

ART DEPARTMENT PROFESSIONAL

Image Processing Power Made Easy

User Manual

SN: 02-220-754-4658

Software and Manual by ASDG, Incorporated





ART DEPARTMENT PROFESSIONAL USER MANUAL JANUARY 20, 1994; NINTH PRINTING

COPYRIGHTS

Copyright © 1990-1994 ASDG, Incorporated. All Rights Reserved.

The Art Department Professional software and its manual are Copyright © 1990-1994 ASDG, Incorporated. All rights reserved. This document may not, in whole or in any part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine readable format without prior consent, in writing, from ASDG, Incorporated.

The JPEG image compression and decompression technology provided with Art Department Professional is done so under license from Handmade Software, Inc. The Installer program is distributed under license from Commodore-Amiga, Inc. The DCTV read/write library is distributed with permission by Digital Creations, Inc. The FARGO Primera dye-sublimation technology is provided under

TRADEMARKS

Art Department Professional is a registered trademark and MorphPlus and Cinemorph are trademarks of ASDG, Incorporated. Kickstart, Workbench, and Advanced Graphics Architecture are trademarks and Amiga is a registered trademark of Commodore-Amiga, Inc. Caligari Broadcast is a trademark of Octree Software, Inc. DCTV is a trademark of Digital Creations, Inc. Deluxe Paint II Enhanced is a trademark of Electronic Arts, Inc. Digi-View is a trademark of NewTek, Inc. FireCracker 24 is a trademark of Impulse Inc. FrameGrabber is a trademark of Progressive Peripherals, Inc. GIF is a trademark of CompuServe Inc. HAM-E is a trademark of Black Belt Systems, Inc. OpalVision and OpalPaint are trademarks of Opal Technology, Ltd. PCX is a trademark of ZSoft Corporation. Primera is a trademark and FARGO is a registered trademark of FARGO Electronics. Professional Page is a trademark of the Gold Disk, Inc. Microsoft Windows is a trademark of Microsoft Corporation. Rendition is a trademark of Numerical Design, Ltd. Sculpt 4D is a trademark of Byte-By-Byte, Inc. Other marks are trademarks of their respective holders.

DISCLAIMER

ASDG, INCORPORATED ("ASDG") MAKES NO WARRANTIES, EITHER EXPRESSED OR IMPLIED, WITH RESPECT TO THE PROGRAM DESCRIBED HEREIN, ITS QUALITY, PERFORMANCE, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. THIS PROGRAM IS SOLD "AS IS." THE ENTIRE RISK AS TO ITS QUALITY AND PERFORMANCE IS WITH THE BUYER. SHOULD THE PROGRAM PROVE DEFECTIVE FOLLOWING ITS PURCHASE, THE BUYER (AND NOT ASDG, THEIR DISTRIBUTORS OR THEIR RETAILERS) ASSUMES THE ENTIRE COST OF ALL NECESSARY REPAIR, CORRECTION, OR SERVICING. IN NO EVENT WILL ASDG BE LIABLE FOR DIRECT, INDIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGE, OR DAMAGES RESULTING FROM LOSS OF USE OR LOSS OF ANTICIPATED PROFITS RESULTING FROM ANY DEFECT IN THE PROGRAM EVEN IF IT HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. SOME LAWS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF IMPLIED WARRANTIES OR LIABILITIES FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES. SO THE ABOVE LIMITATIONS OR EXCLUSION MAY NOT APPLY.

LICENSE

"ENCLOSED PROGRAM" SHALL BE TAKEN TO MEAN THE SOFTWARE ACTUALLY CONTAINED IN THIS PACKAGE AND ANY SUBSEQUENT VERSIONS OR UPGRADES RECEIVED AS A RESULT OF HAVING PURCHASED THIS PACKAGE.

YOU HAVE THE NON-EXCLUSIVE RIGHT TO USE THE ENCLOSED PROGRAM ONLY ON A SINGLE COMPUTER. YOU MAY PHYSICALLY TRANSFER THE PROGRAM FROM ONE COMPUTER TO ANOTHER PROVIDED THAT THE PROGRAM IS USED ON ONLY ONE COMPUTER AT A TIME. HOWEVER, YOU MAY NOT ELECTRONICALLY TRANSFER THE PROGRAM FROM ONE COMPUTER TO ANOTHER OVER A NETWORK. YOU MAY NOT DISTRIBUTE COPIES OF THE PROGRAM OR THE ACCOMPANYING DOCUMENTATION TO OTHERS EITHER FOR A FEE OR WITHOUT CHARGE. YOU MAY NOT MODIFY OR TRANSLATE THE PROGRAM OR DOCUMENTATION. YOU MAY NOT DISASSEMBLE THE PROGRAM OR ALLOW IT TO BE DISASSEMBLED INTO ITS CONSTITUENT SOURCE CODES. YOUR USE OF THE PROGRAM INDICATES YOUR ACCEPTANCE OF THESE TERMS AND CONDITIONS. IF YOU DO NOT AGREE TO THESE CONDITIONS, RETURN THE PROGRAM, DOCUMENTATION, AND ASSOCIATED PERIPHERALS TO THE VENDOR FROM WHOM THIS SOFTWARE WAS PURCHASED.

THIS LICENSE AGREEMENT SHALL BE GOVERNED BY THE LAWS OF THE UNITED STATES OF AMERICA, STATE OF WISCONSIN AND SHALL INURE TO THE BENEFIT OF ASDG, INCORPORATED OR ITS ASSIGNS.

SPECIFIC RESTRICTIONS

IN ACCORDANCE WITH THE COMPUTER SOFTWARE RENTAL ACT OF 1990, THIS SOFTWARE MAY NOT BE RENTED, LENT OR LEASED.

THIS SOFTWARE AND ITS DOCUMENTATION MAY NOT BE PROVIDED BY A "BACKUP SERVICE" OR ANY OTHER VENDOR WHICH DOES NOT PROVIDE AN ORIGINAL PACKAGE AS COMPOSED BY ASDG, INCORPORATED INCLUDING BUT NOT LIMITED TO ALL ORIGINAL DOCUMENTATION, INSERTIONS, REGISTRATION CARDS, AND SOFTWARE.

PRINTED IN THE USA.

Contents

1	Introduction	1
1.1	What is Art Department Professional?	1
1.2	Highlights of Version 2.5	2
1.2.1	Important Notes For Users Of Previous Versions	4
1.3	The ADPro Package	6
1.4	Registration and Product Upgrades	6
1.5	About This Manual	7
1.5.1	Documentation Conventions	9
1.5.2	Example Images	9
1.6	Acknowledgements	10
1.7	About ASDG	10
2	Installation	11
2.1	System Requirements	11
2.2	Memory Requirements	11
2.2.1	Calculating Memory Requirements	12
2.2.2	Limiting Memory Usage	13
2.2.3	Reducing Memory Usage	13

2.3	Installation	14
2.3.1	Novice User Installation	14
2.3.2	Intermediate User Installation	15
2.3.3	Expert User Installation	15
2.4	Serialization	16

3 Basic Concepts 19

3.1	Navigating the User Interfaces	19
3.1.1	User Interface Objects	19
3.1.2	The File Requester	24
3.1.3	The Font Requester	28
3.1.4	The Screen Mode Requester	28
3.1.5	The Chooser	30
3.1.6	The Meter Window	31
3.1.7	The Visual User Interface	32
3.2	Understanding the Image Data Flow	34
3.2.1	Internal Data Types	34
3.2.2	ADPro's Image Buffers	36
3.2.3	ADPro's Building Blocks	37
3.3	Understanding How Loaders Work	37
3.3.1	What a Loader Does Internally	38
3.4	Understanding How Savers Work	39
3.5	Understanding How Operators Work	39
3.6	Understanding Aspect Ratios	40
3.7	Exchanging Data	41

4 Tutorials 43

4.1	Lesson 1. ADPro: File Format Translation	44
4.1.1	Setting Up	44
4.1.2	Selecting the Loader	44
4.1.3	Loading the File	45
4.1.4	Selecting the Saver	45
4.1.5	Saving the File	46
4.1.6	Using the UNIVERSAL Loader	46
4.1.7	Conclusion	47
4.2	Lesson 2. ADPro: Color and Display Adjustments . . .	48
4.2.1	Setting Up	48
4.2.2	Specifying the Amiga Render Screen	48
4.2.3	Rendering and Viewing the Image	48
4.2.4	Using the Dither Modes	49
4.2.5	Adjusting the Balancing	49
4.2.6	Adjusting the Palette	50
4.2.7	Displaying to a Window	51
4.2.8	Conclusion	52
4.3	Lesson 3. ADPro: Using the Palette Editor	53
4.3.1	Setting Up	53
4.3.2	Explicit Ranges	53
4.3.3	Colors Not In A Range	54
4.3.4	Range Commands and Explicit Ranges	54
4.3.5	Range Commands and Implicit Ranges	55
4.3.6	Place Holders	55
4.3.7	Conclusion	56
4.4	Lesson 4. ADPro: Image Compositing	57

4.4.1	Setting Up	57
4.4.2	Loading the Background Image	57
4.4.3	Compositing Using a Transparent Color	57
4.4.4	Compositing Using an Alpha Channel (Matte)	59
4.4.5	Conclusion	59
4.5	Lesson 5. FRED: Batch Image Translation	61
4.5.1	Setting Up	61
4.5.2	Creating a New Sequence	62
4.5.3	Inserting Images Into a Sequence	63
4.5.4	Saving the Sequence	64
4.5.5	Selecting the Images to Process	64
4.5.6	Choosing the File Format to Use	65
4.5.7	Starting the Image Translation	65
4.5.8	Verifying That Images Were Saved Properly	66
4.5.9	Conclusion	66
4.6	Lesson 6. FRED: Batch Image Processing	67
4.6.1	Setting Up	67
4.6.2	Rearranging the Order of Processing	67
4.6.3	Processing the Images	68
4.6.4	Conclusion	68
4.7	Lesson 7. FRED: Creating an ANIM From a Sequence of Images	69
4.7.1	Setting Up	69
4.7.2	Specifying Your Image Sequence	69
4.7.3	Building the List of Scripts	70
4.7.4	Instructing ADPro or MorphPlus to Process Your Images	71

4.7.5	Viewing Your Assembled ANIM	72
4.7.6	Conclusion	72
4.8	Lesson 8. FRED: Creating a Fade Between Two Images	73
4.8.1	Setting Up	73
4.8.2	Creating a Sequence	73
4.8.3	Adding an Image to a Sequence	74
4.8.4	Making Stamps for Images	74
4.8.5	Saving Sequences	74
4.8.6	Creating a Second Sequence	75
4.8.7	Launching the Compositor	75
4.8.8	Specifying Background and Foreground Sequences	76
4.8.9	Specifying the Resulting Sequence	77
4.8.10	Processing the Images	78
4.8.11	Displaying the Fade	78
4.8.12	Conclusion	79
4.9	Lesson 9. Sentry: Automated JPEG Archiving	80
4.9.1	Setting Up	80
4.9.2	Specifying the Directory to Monitor	80
4.9.3	Filtering the Files to Process	81
4.9.4	Specifying How to Process Files	81
4.9.5	Processing JPEG Files	82
4.9.6	Conclusion	83

5 Using ADPro 85

5.1	Overview	85
5.1.1	Starting the Program	85

5.1.2	Displaying Information About ADPro	87
5.1.3	Exiting the Program	87
5.2	Using the Controls	88
5.2.1	The List Interface	88
5.2.2	The Button Interface	90
5.2.3	Floating Module Lists	92
5.2.4	AppWindow, AppIcon, and AppMenu Support	93
5.3	Loading an Image	94
5.3.1	Loader Module	94
5.3.2	Load Orientation	95
5.3.3	Image Compositing	95
5.3.4	Locked Screen Mode and Depth	102
5.4	Saving an Image	102
5.4.1	Saver Module	103
5.4.2	Save Data Types	103
5.5	Operating On an Image	104
5.5.1	Operator Module	104
5.5.2	What Gets Modified?	104
5.6	Displaying an Image	105
5.6.1	Render Screen	105
5.6.2	Number of Colors	107
5.6.3	Dither Type	109
5.6.4	Rendering Controls	111
5.7	Using the Balancing Controls	113
5.7.1	Color Map Basics	114
5.7.2	Red, Green, and Blue Adjustment	115

5.7.3	Brightness Adjustment	116
5.7.4	Contrast Adjustment	116
5.7.5	Gamma Adjustment	117
5.7.6	Balancing Settings Stored In IFF Files	118
5.8	Using the Palette Controls	119
5.8.1	Palette Editing	119
5.8.2	Custom Palette Adjustments	122
5.8.3	Palette Contents	124
5.8.4	Palette Status	126
5.8.5	Palette Accuracy	127
5.9	Printing an Image	127
5.9.1	Preferences-Supported Printers	128
5.9.2	Postscript Printers	128
5.10	Using the Clipboard	128
5.11	Preferences	129
5.11.1	Selecting ADPro's Screen Mode	129
5.11.2	Selecting ADPro's Font	129
5.11.3	Adjusting ADPro's Window Size	129
5.11.4	Setting ADPro's Hot-Key Activation Sequence	130
5.11.5	Customizing ADPro's AppIcon Imagery	130
5.11.6	Loading and Saving Custom Settings	130

6 Loaders 131

6.1	UNIVERSAL	131
6.2	ALPHA	132
6.3	ANIM	134

6.4	BACKDROP	135
6.5	BACKLINE	137
6.6	BMP	138
6.7	CLIPBOARD	138
6.8	CDXL	139
6.9	DPAINT	140
6.10	DPIIE	141
6.11	DV21	141
6.12	FC24	142
6.13	FLC	142
6.14	FRACTAL2000	143
6.14.1	Using the FRACTAL2000 Loader	143
6.14.2	Using the Loader With Mand2000	145
6.14.3	About Mand2000	146
6.15	FRAMEGRABBER	147
6.16	FRAMESTORE	149
6.17	GIF	149
6.18	HAME	149
6.19	ICO	150
6.20	ICON	150
6.21	IFF	151
6.22	IMPULSE	152
6.23	IV24	152
6.24	JPEG	153
6.25	MACPAINT	154
6.26	PAR_PEG	154

6.27	PCX	155
6.28	POINTER	155
6.29	QRT	156
6.30	SCREEN	157
6.31	SCULPT	158
6.32	TEMP	159
6.33	VLAB	160
6.34	YUVN	160
6.35	_LoadNShowFC24	160
6.36	_LoadNShowOV	161
6.37	_LoadNShowPICASSO	161
6.38	_LoadNShowRETINA	161
6.39	_VT_Grab	161

7 Savers 163

7.1	A2410	163
7.2	ANIM	166
7.3	BMP	168
7.4	CDXL	169
7.5	CLIPBOARD	171
7.6	DPAINT	171
7.7	DPIIE	173
7.8	FC24	173
7.9	FRAMEBUFFER	177
7.10	FRAMESTORE	177
7.11	GIF	179

7.12	HAME	179
7.13	HARLEQUIN	181
7.14	ICON	183
7.15	IFF	184
7.16	IMPULSE	184
7.17	IV24	185
7.18	JPEG	187
7.19	OPALVISION	188
7.20	PCX	189
7.21	PICASSO	190
7.22	POSTSCRIPT	191
7.23	PREFPRINTER	198
7.23.1	Setting the Preferences Paper Type	198
7.23.2	Specifying the Print Destination	199
7.23.3	Specifying the On-Screen Dimensions	199
7.23.4	Setting the Printer Driver and Its Attributes	200
7.23.5	Setting the Dither Type	201
7.23.6	Setting the Print Aspect	203
7.23.7	Specifying the Output Settings	204
7.23.8	Using the Color Correction Controls	206
7.23.9	Controlling the FARGO Primera Dye-Sublimation Printer	206
7.23.10	Initiating a Print	208
7.24	QRT	208
7.25	RESOLVER	208
7.26	RETINA	210
7.27	SCULPT	211

7.28	SEPARATE	211
7.28.1	Using ADPro Separations With Professional Page	214
7.29	TEMP	215
7.30	VT_BUFFER	216
7.31	_MultiCopyPREFPRINTER	216
7.32	_SCULPTWithDimen	216
7.33	_VT_Display	217

8 Operators 219

8.1	Antique	219
8.2	Apply_Map	219
8.3	Blur	220
8.4	Broadcast_Limit	221
8.5	Collapse	222
8.6	Color_To_Gray	224
8.7	Colorize	225
8.8	Convolve	227
8.9	Crop_Image	228
8.10	Crop_Visual	230
8.11	DCTV	231
8.12	Define_Pxl_Aspect	231
8.13	DeInterlace	232
8.14	Displace_Pixel	233
8.15	Dynamic_Range	234
8.16	Gray_To_Color	234
8.17	Halve	235

8.18	Horizontal_Flip	235
8.19	Hist_Equalization	235
8.20	Intensity_Range	236
8.21	Interlace	237
8.22	KillTemp	237
8.23	Line_Art	238
8.24	Median_Filter	238
8.25	Mosaic	239
8.26	Negative	240
8.27	OpalPaint	240
8.28	Pattern	241
8.29	Polar_Mosaic	242
8.30	Rectangle	243
8.31	Rectangle_Visual	244
8.32	Rem_Isolated_Pxls	245
8.33	Rendered_To_Raw	247
8.34	Roll	248
8.35	Rotate	248
8.36	Saturation	251
8.37	Scale	251
8.38	Sim_Print	254
8.39	Text_Visual	255
	8.39.1 Example Usage	260
8.40	Tile	261
8.41	Tile_Visual	265
8.42	TPort_Controller	266

8.43	Twirl	266
8.44	Vertical_Flip	269
8.45	_ColorCharcoal	269
8.46	_DoubleSize	269
8.47	_Emboss	269
8.48	_Fresco	270
8.49	_Highlight	270
8.50	_Mirror	270
8.51	_OilPaint	270
8.52	_RemapNRerender	270
8.53	_RotateImage	270
8.54	_ScaleToPixelAspect	270
8.55	_Solarize	271

9 Displays 273

9.1	Amiga	273
9.1.1	Render Controls	274
9.1.2	Display Restrictions	274
9.2	EGS	275
9.2.1	Render Controls	276
9.3	FC24	276
9.3.1	Render Controls	277
9.4	OpalVision	277
9.4.1	Render Controls	277
9.5	Picasso	277
9.5.1	Render Controls	278

9.6	Retina	278
9.6.1	Render Controls	278
9.7	Window	279
9.7.1	Render Controls	279

10 FRED 281

10.1	Overview	281
10.1.1	Starting the Program	282
10.1.2	Displaying Information About FRED	283
10.1.3	Exiting the Program	283
10.2	Sequences, Cells, and Frames	283
10.2.1	Creating Sequences	283
10.2.2	Adding Images to a Sequence	284
10.2.3	Storing and Retrieving Sequences	285
10.2.4	Switching Between Sequences	286
10.2.5	Differentiating Cells and Frames	286
10.3	Cell Selection and Editing	286
10.3.1	Selecting and Unselecting Cells	286
10.3.2	Arranging Cells	287
10.3.3	Loading Images Into ADPro	287
10.4	Cell Information	288
10.4.1	Selecting the Cell Text	288
10.4.2	Setting the Duration	289
10.5	Animated Stamps	290
10.5.1	Creating Stamps	290
10.5.2	Animating the Stamps	290

10.6	Sequence Processing	291
10.6.1	Pre and Post Processing	293
10.6.2	Contents of a FREDScript	294
10.7	AnimOps Processing	296

11 AnimOps 297

11.1	Cinemorph	297
11.1.1	Process Controls	298
11.1.2	Input Controls	299
11.1.3	Conversion Controls	299
11.1.4	Output Controls	299
11.2	Compositor	300
11.2.1	Projects	300
11.2.2	Components	301
11.2.3	Results	303
11.3	Time.Stretch	304
11.3.1	Main Control Panel	305
11.3.2	Extra Options Control Panel	306

12 ARexx Interface 309

12.1	General Information	309
12.1.1	Addressing ADPro	309
12.1.2	Getting Results of Commands	310
12.1.3	Invoking Commands	311
12.1.4	Understanding the General Rules	312
12.2	Launching ARexx Scripts	313
12.2.1	Keyboard Macros	313

12.2.2	Pseudo Modules	315
12.2.3	User Commands	315
12.2.4	Execute ARexx Script	316
12.3	ARexx Commands Cross Reference	316
12.4	ADProScripts	319
12.5	FREDScripts	323
12.5.1	FREDOperators	323
12.5.2	FREDRenderers	329
12.5.3	FREDSavers	330
12.5.4	FREDFunctions	334

13 Reference 339

13.1	ADPro Preferences	340
13.1.1	Public Screen Name	340
13.1.2	Tool Type Options	340
13.1.3	Command Line (Shell) Arguments	342
13.2	ADPro Keyboard Shortcuts	344
13.3	ADPro Menus	346
13.3.1	Project Menu	346
13.3.2	Edit Menu	347
13.3.3	Loaders Menu	347
13.3.4	Savers Menu	347
13.3.5	Operators Menu	347
13.3.6	Display Menu	348
13.3.7	Settings Menu	348
13.3.8	User Menu	349

13.4	General Controls and Information	350
13.4.1	Version Number	350
13.4.2	Serial Number	350
13.4.3	Image Buffer Size	350
13.4.4	Exiting ADPro	350
13.4.5	Specifying ADPro's Screen	351
13.4.6	Specifying on Which Public Screen to Open ADPro	351
13.4.7	Bringing the ADPro Screen to Front	352
13.4.8	Pushing the ADPro Screen Behind	352
13.4.9	Bringing a Public Screen to Front	352
13.4.10	Specifying the Type of ADPro's Interface	352
13.4.11	Specifying the Size of ADPro's Interface	353
13.5	Balancing Controls	353
13.5.1	Red Adjustment	353
13.5.2	Green Adjustment	354
13.5.3	Blue Adjustment	354
13.5.4	Brightness Adjustment	355
13.5.5	Contrast Adjustment	355
13.5.6	Gamma Adjustment	355
13.6	Palette Controls	356
13.6.1	Palette Width	356
13.6.2	Palette Status	357
13.6.3	Total Number of Colors	357
13.6.4	Number of Colors Used	358
13.6.5	Offset Color Zero	358
13.6.6	Custom Palette Values	359

13.6.7	Palette Contrast	359
13.6.8	Palette Sort Order	360
13.6.9	Individual Palette Contents	360
13.6.10	Multiple Palette Contents	361
13.6.11	Setting Pixel Colors	361
13.6.12	Getting Pixel Colors	361
13.6.13	Getting the Workbench Palette	362
13.6.14	Loading a Palette	362
13.6.15	Saving a Palette	363
13.6.16	The Wrong Way to Control the Palette	363
13.7	Other ARexx Commands	364
13.7.1	Last Saved Image	364
13.7.2	Last Loaded Image	364
13.7.3	Pausing a Specified Time	365
13.7.4	Getting a String From the User	365
13.7.5	Getting an Integer Number From the User . .	366
13.7.6	Getting a Floating Point Number From the User	366
13.7.7	Getting a Filename From the User	367
13.7.8	Getting a List of Filenames From the User . .	368
13.7.9	Getting a Directory Name From the User . . .	369
13.7.10	Getting a Screen Mode From the User	370
13.7.11	Retrieve the Entries In Internal Lists	371
13.7.12	Select From a List of Items	371
13.7.13	Display a Message	372
13.7.14	The Notify Requester	372
13.7.15	The Two-Choice Response Requester	373

13.7.16	The Multiple-Choice Response Requester . . .	373
13.7.17	Loading a Settings File	373
13.7.18	Saving a Settings File	374
13.7.19	Obtaining Exclusive Use of ADPro	374
13.7.20	Releasing Exclusive Use of ADPro	374
13.8	Loaders	375
13.8.1	Setting the Load Format	375
13.8.2	Invoking the Current Loader	375
13.8.3	Invoking a Loader	376
13.8.4	Launching a Loader	376
13.8.5	Compositing Options	376
13.8.6	Load Orientation	379
13.8.7	Load Mode	379
13.8.8	ALPHA	380
13.8.9	ANIM	380
13.8.10	BACKDROP	381
13.8.11	BACKLINE	382
13.8.12	BMP	383
13.8.13	CDXL	383
13.8.14	CLIPBOARD	384
13.8.15	DPAINT	384
13.8.16	DPIIE	385
13.8.17	DV21	386
13.8.18	FC24	386
13.8.19	FLC	386
13.8.20	FRACTAL2000	387

13.8.21	FRAMEGRABBER	387
13.8.22	FRAMESTORE	388
13.8.23	GIF	388
13.8.24	HAME	389
13.8.25	ICO	389
13.8.26	ICON	389
13.8.27	IFF	389
13.8.28	IMPULSE	390
13.8.29	IV24	390
13.8.30	JPEG	391
13.8.31	MACPAINT	391
13.8.32	PAR_PEG	392
13.8.33	PCX	392
13.8.34	POINTER	393
13.8.35	QRT	393
13.8.36	SCREEN	393
13.8.37	SCULPT	394
13.8.38	TEMP	395
13.8.39	UNIVERSAL	395
13.8.40	VLAB	395
13.8.41	YUVN	395
13.9	Savers	397
13.9.1	Setting the Save Format	397
13.9.2	Invoking the Current Saver	397
13.9.3	Invoking a Saver	398
13.9.4	Launching a Saver	398

13.9.5	A2410	398
13.9.6	ANIM	400
13.9.7	BMP	401
13.9.8	CDXL	401
13.9.9	CLIPBOARD	403
13.9.10	DPAINT	403
13.9.11	DPIIE	404
13.9.12	FC24	404
13.9.13	FRAMEBUFFER	406
13.9.14	FRAMESTORE	406
13.9.15	GIF	407
13.9.16	HAME	407
13.9.17	HARLEQUIN	408
13.9.18	ICON	410
13.9.19	IFF	410
13.9.20	IMPULSE	411
13.9.21	IV24	411
13.9.22	JPEG	412
13.9.23	OPALVISION	413
13.9.24	PCX	414
13.9.25	PICASSO	415
13.9.26	POSTSCRIPT	416
13.9.27	PREFPRINTER	416
13.9.28	QRT	419
13.9.29	RESOLVER	419
13.9.30	RETINA	421

13.9.31	SCULPT	422
13.9.32	SEPARATE	422
13.9.33	TEMP	425
13.9.34	VT_BUFFER	425
13.10	Operators	427
13.10.1	Setting the Operate Format	427
13.10.2	Invoking the Current Operator	427
13.10.3	Invoking an Operator	427
13.10.4	Launching an Operator	428
13.10.5	Antique	428
13.10.6	Apply_Map	428
13.10.7	Blur	428
13.10.8	Broadcast_Limit	429
13.10.9	Collapse	430
13.10.10	Color_To_Gray	431
13.10.11	Colorize	431
13.10.12	Convolve	432
13.10.13	Crop_Image	433
13.10.14	Crop_Visual	434
13.10.15	DCTV	434
13.10.16	Define_Pxl_Aspect	434
13.10.17	DeInterlace	436
13.10.18	Displace_Pixel	436
13.10.19	Dynamic_Range	436
13.10.20	Gray_To_Color	437
13.10.21	Halve	437

13.10.22	Hist_Equalization	438
13.10.23	Horizontal_Flip	438
13.10.24	Intensity_Range	438
13.10.25	Interlace	438
13.10.26	KillTemp	439
13.10.27	Line_Art	439
13.10.28	Median_Filter	439
13.10.29	Mosaic	440
13.10.30	Negative	440
13.10.31	OpalPaint	441
13.10.32	Pattern	441
13.10.33	Polar_Mosaic	442
13.10.34	Rectangle	443
13.10.35	Rectangle_Visual	444
13.10.36	Rem_Isolated_Pxls	444
13.10.37	Rendered_To_Raw	444
13.10.38	Roll	444
13.10.39	Rotate	445
13.10.40	Saturation	446
13.10.41	Scale	446
13.10.42	Sim_Print	447
13.10.43	Text_Visual	448
13.10.44	Tile	452
13.10.45	Tile_Visual	452
13.10.46	TPort_Controller	452
13.10.47	Twirl	453

13.10.48 Vertical_Flip	454
13.11 Displays	455
13.11.1 Number of Colors in Rendered Image	455
13.11.2 Rendering an Image	455
13.11.3 Displaying an Image	456
13.11.4 Ending the Display of the Image	456
13.11.5 Render Mode Status	456
13.11.6 Render Depth Status	456
13.11.7 Availability of Render Modes	457
13.11.8 Removing the Raw Image Data	457
13.11.9 Removing the Rendered Image Data	457
13.11.10 Closing the Render Screen	458
13.11.11 Screen Type	458
13.11.12 Setting the Render Screen Mode	459
13.11.13 Dither Selection	460
13.11.14 Dither Amount	460
13.11.15 Image Width	461
13.11.16 Image Height	461
13.11.17 Determining Currently Available Data	461
13.11.18 Pixel Aspect	462
13.11.19 Pixel Resolution	462
13.11.20 Amiga	463
13.11.21 EGS	463
13.11.22 FC24	463
13.11.23 OpalVision	463
13.11.24 Picasso	463

13.11.25 Retina	463
13.11.26 Window	464
13.12 FRED Preferences	465
13.12.1 Public Screen Name	465
13.12.2 Tool Type Options	465
13.12.3 Command Line (Shell) Arguments	465
13.13 FRED Keyboard Qualifiers	465
13.14 FRED Menus	466
13.14.1 Project Menu	466
13.14.2 Settings Menu	466
13.14.3 Tools Menu	466
13.14.4 AnimOps Menu	467
13.14.5 Edit Menu	467
13.14.6 Animation Menu	468
13.14.7 Scripts Menu	468

A Troubleshooting 469

A.1 Technical Support	469
---------------------------------	-----

B Hints and Tips 473

B.1 ADPro	473
B.1.1 Anti-Aliasing	474
B.1.2 Approximating Charcoal Drawings	474
B.1.3 Avoiding the Creation of Corrupt IFF Files	475
B.1.4 Caching the Loader, Saver, and Operator Modules	475
B.1.5 Changing Input Field Values	476

B.1.6	Compositing Tips	476
B.1.7	Creating Better Gradated Fills	477
B.1.8	Creating Drop Shadows	477
B.1.9	Eliminating Stray Dots	478
B.1.10	Example FG and FC24 Use	478
B.1.11	Focused Color Picking	480
B.1.12	Grayscale Balancing	480
B.1.13	Merging a Color Image Into a Grayscale Image	481
B.1.14	Mixing Picked and Computed Colors	481
B.1.15	Multiple ADPro Configurations From The Workbench	482
B.1.16	Parsing Strings in ARexx	483
B.1.17	Reading and Writing Toaster Files	483
B.1.18	Regionalized Processing	483
B.1.19	Scaling as an Anti-Aliasing Tool	484
B.1.20	Starting ADPro Via ARexx	484
B.1.21	Suggested PostScript Parameters For Color Work	486
B.1.22	Time Lapse and Motion Blur	486
B.1.23	Turning Low Color Images Into Higher Res Grayscale	487
B.1.24	Working With Moving Video	487
B.1.25	Working With Pictures Which Are Too Large .	487

C Third-Party Developer Support 489

D PREFPRINTER Dither Type Samples 491

E Additional Utilities 497

E.1	Sentry	497
E.1.1	Starting Sentry	497
E.1.2	Configuring Sentry	498
E.1.3	Projects	499
E.1.4	Directory Notification	500
E.1.5	Command List	500
E.1.6	Activation and Monitoring	502
E.1.7	Commodity Support	503
E.1.8	Network and File System Issues	504
E.1.9	ARexx Issues	505
E.2	Splitz & Joinz	505
E.2.1	Amiga Set-Up and Usage	506
E.2.2	Macintosh Set-Up and Usage	508
E.2.3	PC/MS-DOS Set-Up and Usage	509
E.2.4	Windows Set-Up and Usage	510
E.3	ZapDPI	511
E.4	ReleaseNotes	512

Index

List of Figures

2.1	The ADPro Serialization panel.	16
3.1	An example menu strip hierarchy.	20
3.2	The ASL File Requester.	25
3.3	The ASL Font Requester.	28
3.4	The Screen Mode requester used in ADPro.	29
3.5	An example of a Chooser list.	30
3.6	The meter window showing an operation 50% complete.	32
3.7	The Crop_Visual control screen, a typical user of the Visual User Interface.	33
3.8	Data flow within ADPro.	35
3.9	The Image Information area, showing the existence of raw and rendered color image data for a 640 x 400 image.	36
3.10	ADPro's Building Blocks.	37
3.11	Comparison of various aspect ratios.	40
5.1	The main ADPro controls in the List Interface layout.	89
5.2	The main ADPro controls in the Button Interface layout.	91
5.3	The Image Compositing screen and control panel.	96
5.4	Dither Amount control panel.	110

5.5	Render screen scrolling via numeric keypad.	112
5.6	The Balancing control panel.	114
5.7	A linear (neutral) color map.	115
5.8	A color map showing an increase in brightness.	116
5.9	A color map showing a decrease in contrast.	117
5.10	A color map showing an example of gamma correction.	118
5.11	The Palette control panel.	119
5.12	The 256 color Palette Editor.	120
6.1	The ANIM loader control panel.	134
6.2	The BACKDROP loader control panel.	136
6.3	The BACKLINE loader control panel.	137
6.4	The CDXL loader control panel.	139
6.5	The DPaint IV AGA loader control panel.	140
6.6	The FC24 loader control panel.	142
6.7	The FLC loader control panel.	143
6.8	The FRACTAL2000 loader control panel.	143
6.9	The FRAMEGRABBER loader control panel.	147
6.10	The ICON loader control panel.	150
6.11	The IV24 loader control panel.	153
6.12	The JPEG loader control panel.	153
6.13	The PAR_PEG loader control panel. For still frames, the Frame: control is not available.	155
6.14	The POINTER loader control panel.	156
6.15	The SCREEN loader control panel.	157
6.16	The SCULPT loader control panel.	158
7.1	The A2410 saver control panel.	164

7.2	The ANIM Saver control panel. The Count Frames button has already been selected (Number of Frames: is not zero), causing it to be ghosted.	166
7.3	The CDXL Saver control panel for new files.	170
7.4	The CDXL Saver control panel for existing files.	170
7.5	The DPaint IV AGA saver control panel.	171
7.6	The first of two control panels for the FireCracker 24 saver.	174
7.7	The second of two control panels for the FireCracker 24 saver.	176
7.8	The FRAMESTORE saver control panel.	178
7.9	The HAME Saver control panel.	179
7.10	The ACS HARLEQUIN saver control panel.	182
7.11	The ICON saver control panel.	183
7.12	The IMPULSE saver control panel.	185
7.13	The IV24 saver control panel.	186
7.14	The JPEG saver control panel.	187
7.15	The OPALVISION saver control panel.	188
7.16	The primary POSTSCRIPT saver control panel.	192
7.17	Various PostScript metrics.	194
7.18	The second POSTSCRIPT saver control panel.	195
7.19	The third POSTSCRIPT saver control panel.	196
7.20	The PREFPRINTER control screen. Note the print direction arrow in the middle of the first page to be printed.	199
7.21	Various margins and measures associated with the PREFPRINTER saver.	207
7.22	The RESOLVER saver control panel.	209
7.23	The SEPARATE saver control panel.	212

7.24	The VT_BUFFER saver control panel.	216
8.1	The Blur operator control panel.	220
8.2	The Broadcast_Limit operator control panel.	221
8.3	The Collapse operator control panel.	223
8.4	The Color_To_Gray operator control panel.	225
8.5	The Colorize operator control panel.	226
8.6	The Convolve operator control panel.	227
8.7	The left half of this ray tracing has been sharpened using the Convolve operator.	229
8.8	The Crop_Image operator control panel.	229
8.9	The Crop_Visual operator control screen.	230
8.10	The Define_Pxl_Aspect operator control panel.	232
8.11	The Displace_Pixel operator control panel.	233
8.12	The Dynamic_Range operator control panel.	234
8.13	The Median_Filter operator control panel.	239
8.14	The Mosaic operator control panel.	240
8.15	The Pattern operator control panel.	241
8.16	The Polar_Mosaic operator control panel.	243
8.17	The Rectangle operator control panel.	244
8.18	The Rectangle_Visual control screen and panel showing how a completely filled rectangle is represented.	246
8.19	The Rectangle_Visual control screen and panel show- ing how a rectangle with a non-zero thickness (but not completely filled) is represented.	246
8.20	The Roll operator control panel.	248
8.21	The Rotate operator control screen.	249
8.22	Components of the Rotate Circle.	249

8.23	The Saturation operator control panel.	251
8.24	The Scale operator control panel.	253
8.25	The Sim.Print operator control panel.	254
8.26	The Text_Visual operator control screen and main panel.	256
8.27	The Tile operator control panel.	262
8.28	A tiling showing no skew.	263
8.29	A tiling showing horizontal skew.	263
8.30	A tiling showing vertical skew.	264
8.31	A tiling showing no skew in which the tiled area evenly fits in the larger bit-map.	265
8.32	The Tile_Visual operator control screen and panel. . . .	266
8.33	The Twirl operator control screen.	267
8.34	Components of the Twirl Circle.	268
10.1	Components of a FRED sequence window.	284
10.2	The Set Text Options control panel.	288
10.3	The Change Duration control panel.	289
10.4	The Image Information control panel.	290
10.5	The Invoke ADPro control panel.	292
11.1	Cinemorph's main control panel.	298
11.2	The Compositor AnimOp's Components control panel.	301
11.3	The Compositor AnimOp's Results control panel. . . .	303
11.4	The Time.Stretch AnimOp's main control panel.	305
11.5	The Time.Stretch AnimOp's Extra Options control panel.	306
E.1	Sentry's main control panel.	498
E.2	Sentry's Command Specification control panel.	501

E.3	Sentry's status panel.	502
E.4	Sentry's cell information control panel.	503

List of Tables

5.1	Types of image data that can be merged together. . . .	96
5.2	Key sequences to position the source (foreground) image box.	97
5.3	How mix level affects the new composite image.	98
5.4	RGB values for the transparent color preset buttons. .	99
5.5	Render screen scrolling controls.	112
6.1	A list of some of the file formats detectable by the UNIVERSAL loader, the filename extensions recognized as hints, and indication of which modules come standard with a particular package.	133
7.1	Resolutions supported by the A2410 and the crystals required to implement them.	164
7.2	The OPALVISION saver keyboard equivalents.	189
7.3	Dither mask size versus effective color resolution. . . .	201
9.1	Color modes and other information. The mixtures of color and display modes marked “A” are possible but require non-standard hardware or an AGA-equipped machine. ADPro allows these modes to be computed but will display them only if the appropriate enhanced hardware is available.	275

12.1	Naming conventions for function key macros.	313
12.2	Naming conventions for numeric keypad macros.	314
13.1	TIGA screen IDs which may be used as arguments to the B_SIZE parameter.	399
13.2	SCREEN_TYPE mask values for the OPALVISION saver.	413
13.3	Values used in constructing a TEXT_STYLE	451
13.4	Mask values for setting the screen type.	458
13.5	Dither methods and their identifiers.	460
A.1	Methods of Contacting ASDG.	470
B.1	Suggested angles and densities for color work run on the Linotronic L300 or L330.	486
B.2	Mix percentages to produce a time lapse effect.	486
E.1	Sentry's Environment variables.	501

Chapter 1

Introduction

This chapter describes the overall Art Department Professional package. After reading it, you will understand:

- what is Art Department Professional
- how to use the manual
- what disks and other materials are included in the package
- why it is important to register your copy of Art Department Professional
- how to receive technical support

1.1 What is Art Department Professional?

Art Department Professional (ADPro) is an integrated set of powerful image processing tools which facilitate the creation of high quality imagery on the Commodore Amiga personal computer. ADPro's main qualities include:

- having device and format independent input, output, and processing capabilities. This means that it can read and write nearly any file format, performing any format conversion required. Also, it can control nearly any form of color input or output device, such as color printers, scanners, film recorders, and digitizers.
- programmability using the ARexx programming language, making it ideal for use in large projects, such as the creation of CDTV/CD32 applications, animations, presentations, and large publishing works.

- display support for many graphics devices, including the A2410, DCTV, any EGS-compatible board, FireCracker 24, HAM-E, Harlequin, IV24, OpalVision, Picasso II, Retina/Retina Z-III, and the Video Toaster's frame buffers.
- coming from the company which pioneered color image processing on the Amiga. A company whose only focus is the development of color imaging solutions. And, a company which has been in continuous business in the Amiga market since the computer was introduced in 1985.

1.2 Highlights of Version 2.5

Following is a brief outline of some of the highlights of version 2.5 of ADPro for the benefit of our continuing customers:

- A completely redesigned main user interface, taking full advantage of the Amiga's standard graphical system of windows, gadgets, menus, and screen modes! ADPro's controls are now in a window that can reside on the Workbench screen, its own custom public screen, or any other public screen. You can now run ADPro on FRED's screen for easy access to all your controls.

All new or revised windows sporting this new look are fully font-sensitive and abide by the Amiga's Style Guide for user interfaces, thereby decreasing the time required to learn the new user interface. You can even select the font to use for its controls when running on its own screen.

Extensive keyboard shortcuts have been added as well. They have been designed to be consistent throughout the program, so that you don't have to learn a new set of shortcuts when moving from module to module.

Almost all of the loader, saver, and operator control panels have been updated to the new look as well.

- A choice of user interfaces. You can run ADPro in either Button Interface mode, which gives you easy access to a few commonly-used modules (which are user-definable), or in List Interface mode, which gives you immediate access to all of your modules. In Button Interface mode, you can use the "tear-off" loader, saver, operator, and user commands list windows to get the best of both worlds; these "tear-off" lists are available in the List mode as well.

You can also minimize the ADPro window to a small size and move it to a corner of your screen. This gives you more room to work with other applications, but allows you to tap into ADPro's image processing horsepower with a simple click of the mouse or press of a few keys.

Commodity support has been added as well, so that you can call up ADPro with a user-definable hot-key sequence.

When running ADPro on the Workbench screen, its window becomes an AppWindow. You can drop files on this window and have it launch an ARexx script. AppIcon and AppMenu supported has also been added.

- Modular display options! You can now render your images from the main window (using the **Redisplay** and **Execute** commands) to any supported Amiga screen, to graphics boards (such as the FireCracker 24, Retina/Retina Z-III, Picasso II, or any EGS-compatible board), or into a window on any public screen! Being a modular system, third-party developers can make their own display modules, extending the usefulness of ADPro.

Saver modules for graphics boards are also included as an alternative display option and for added control.

- Advanced compositing controls. You can now specify an alpha channel file in a non-IFF format; the foreground image to be composited doesn't have to be IFF as well. Alpha channel files do not have to be grayscale either—color images will be converted automatically.

A range of colors, in addition to just one color, can be specific as being transparent. Now you don't have to know the exact RGB color value, which is especially useful for those images with areas that are "somewhat" black, but not 0-0-0.

When compositing an image, a visual screen is opened. Extensive keyboard control over the position of the incoming image is also provided. Embedded alpha channels in files can be extracted, used, or ignored.

- Easy access to your ARexx scripts via User Commands. Now you don't have to rename your scripts to use them from ADPro. They are available either as buttons or in a list from the main window.

You can also execute any ARexx script by selecting it from a file requester.

- Support for CDXL, FLC, ICO, and ICON files. The UNIVERSAL loader has been updated to support these, as well as the new formats (ALIAS, SGI, and WAVEFRONT) in the Professional Conversion Pack expansion product.
- Support for the FARGO Primera dye-sublimation (photo-realistic) printer. The revised PREFPRINTER interface has been updated to output to either the Preferences printer (having a printer driver that supports strip printing) or Primera dye-sublimation printer. The Primera's normal wax thermal output is handled by the Primera printer driver via Preferences.
- FRED has been improved as well, including a more Style Guide compliant user interface, enhanced AnimOps, and many more pre-written FREDScripts for your enjoyment.
- Various new and enhanced features, such as the ability to lock to a specific aspect during scaling, new ARexx commands, locking the screen

mode and depth so the next loaded image will use the previous values, limiting the screen modes that you can select to those supported by your graphics hardware, the ability to load and save your working environment, easy access to common modules (UNIVERSAL loader, IFF saver, PREFPRINTER saver, and CLIPBOARD loader and saver), and much more!

Again, these are just highlights of the new capabilities in ADPro version 2.5. With the original ADPro, we earned a dominant place in the color imaging landscape. With the release of version 2.5 of ADPro, we believe we will earn that distinction again. Your continuing support has made this possible.

1.2.1 Important Notes For Users Of Previous Versions

This version of ADPro incorporates many cosmetic changes which may take some time to get acquainted with, especially if you've been a long-time user of ADPro. This section is intended to help get you up-and-running as quick as possible by pointing how the major changes to the program.

User Interface

As you probably have already found out, ADPro's controls are now in a window which can appear on any public screen, instead of the old low resolution, non-interlace custom screen. There are now menus, standard gadgets and buttons, and standard file requesters. Not all modules have been updated, but a fair number of them do sport the new look.

Also, extensive keyboard control has been added. These keyboard shortcuts are identified by an underscore ("_") under the shortcut. One keyboard shortcut which has been *removed* is **Shift + Return** to accept the current changes to a module, especially an operator. All of the unmodified modules should still use this shortcut—the new ones should be affected.

AppWindow, AppMenu, and AppIcon support, the ability to run ADPro as a window on any public screen or on its own screen, hot-key support, customizable interface, custom font support, minimized mode, and better support for multiple configurations (settings)—these options let you work with ADPro the way you want them to.

Easy Access To Common Modules

Many of the common modules—UNIVERSAL loader, IFF saver, PREF-PRINTER saver, and CLIPBOARD loader and saver—have been given special menu items. They are “disguised” as the more common looking **Open...**, **Save As...**, **Print As...**, **Paste**, and **Copy**, respectively. These functions were put into the menus to be more compliant with other applications, as well as to make it more apparent that ADPro can do these operations.

We have also added “favorite” module buttons and menus for those modules which you use most-often or that you want easy access to. This gives somewhat the same functionality as the **MRU** button in the old Chooser lists. In addition to these “favorite” buttons, you can open floating module lists to get Chooser-like lists that can stay open all the time.

Modular Rendering Options

The old Screen Controls buttons have been replaced with the **Set Render Screen** requester. Now, you select the screen mode from the list of standard Amiga modes. Number of colors and overscan settings are also selectable from this requester. The dither type and amount buttons are now as menu items and subitems under the **Display** menu.

We have gone to a standard-looking screen mode requester not only to gain support for all modes available in the Amiga’s Display Database, but to also allow for the direct rendering to third-party graphics boards and even to windows on any public screen. Now, selecting a rendering “destination” only requires that you learn how to use the Set Render Screen controls.

Compositing Control

If you use ADPro to compose images together, you will welcome the enhancements we’ve made to the compositing controls. Now, instead of a small window on a small screen, compositing takes place on a high resolution visual screen, similar to those used by the visual operators.

Positioning tools are also improved with the addition of keyboard control, so that you can “nudge” your foreground image in varying increments; aligning the edges of your images is also made trivial with these keyboard shortcuts. You can still drag the box with the mouse, but you don’t have to be *within* the box to drag it.

The extension of alpha channel support to non-IFF grayscale images will be a boon to users needing easy tools. Transparency range support should also be a benefit.

Settings Files

ADPro's settings are now stored in a file called "ADPro.prefs" instead of "ADProDefaults". Also, there is a specific order in which ADPro will search for this file (discussed in Chapter 5).

In addition, the tool types and Shell arguments have been made more Style Guide-compliant (e.g., **SETTINGS** in addition to **DEFAULTFILE**). The old options are still supported. However, all the general descriptions in this manual will reference the new options.

1.3 The ADPro Package

The ADPro package comes complete with the following items:

- Six (6) Art Department Professional disks
- This Art Department Professional User Manual
- Registration Card

If you find that you are missing one or more of the above items, please contact the place where you purchased ADPro and ask to exchange your copy with a complete package. If they cannot offer you an exchange, please call us (our phone number is listed in Appendix A) and we will be happy to send you a complete package.

If you find that your version of ADPro does not have a serial number, please contact us by phone and we will send you a properly serialized program.

1.4 Registration and Product Upgrades

ASDG treats the support of its products very seriously. We continually improve our products and, from time to time, make these improvements available in the form of product upgrades.

NOTE It is absolutely imperative that you fill out and return the registration card you received with this product. If you don't, we won't be able to reach you with information about upgrades to ADPro. Also, technical support will **not** be rendered to anyone who is not in our registered user database. So, please, send in your registration card.

When filling out the registration card, be sure to include the serial number of your copy of ADPro. ADPro's serial number can be found in the About panel, displayed by selecting the **About...** menu item under the **Project** menu. The registration card must include the serial number to be valid.

Note that if you take advantage of an upgrade offer (i.e., move from one version of ADPro to another), the new version you receive is considered to be exactly the same software as covered by your license agreement. Therefore, you may not sell, lend, lease, or give away your old version as this would be an infringement upon our copyright.

You can sell your copy of ADPro only if: you destroy all copies in your possession, deliver the original disks and manuals to the buyer, and inform ASDG of the transfer of ownership so that the new owner may receive technical assistance from us.

1.5 About This Manual

The documentation for ADPro is divided into several parts. These are:

- | | |
|------------------|---|
| Chapter 1 | Introduction
Contains introductory explanations about the manual, the software, and technical support. |
| Chapter 2 | Installation
Contains general information and helpful tips on installing the Art Department Professional software. |
| Chapter 3 | Basic Concepts
Contains basic concepts you should be familiar with before you can try out the tutorials and use ADPro. |
| Chapter 4 | Tutorials
Contains a few tutorials that will help you become more familiar with Art Department Professional and using its controls to create the desired results. |
| Chapter 5 | Using ADPro
Contains the general explanation of how to use the ADPro program. |
| Chapter 6 | Loaders
Contains the main reference for all ADPro loaders. |
| Chapter 7 | Savers
Contains the main reference for all ADPro savers. |

Chapter 8	Operators
	Contains the main reference for all ADPro operators.
Chapter 9	Displays
	Contains the main reference for all ADPro display modules.
Chapter 10	FRED
	Contains the main reference for the FRED program.
Chapter 11	AnimOps
	Contains the main reference for all FRED AnimOps.
Chapter 12	ARexx Interface
	Contains general information about the ARexx interface of the ADPro program. Actual ARexx commands are listed in the master reference chapter. Descriptions of all the ADPro and FRED ARexx scripts are also included in this chapter.
Chapter 13	Reference
	Contains a master reference of all of ADPro's control panel commands, keyboard equivalents, menu items, and ARexx arguments.
Appendix A	Troubleshooting
	Contains explanations and possible solutions to several problems and questions you might encounter while using ADPro.
Appendix B	Hints and Tips
	Contains some helpful hints and tips for getting the most use out of ADPro.
Appendix C	Third-Party Developer Support
	Contains information on how to create expansion modules for ADPro.
Appendix D	PREFPRINTER Dither Type Samples
	Contains sample printouts of the dither types available with the PREFPRINTER saver module.
Appendix E	Additional Utilities
	Contains descriptions of the miscellaneous utilities included with the ADPro package.
Index	Contains an index for the manual.

The manual is primarily structured as a user manual for you to read from chapter to chapter. However, it can also be used as a reference manual for the program, by using the included master reference in Chapter 13. In the early stages of learning the program, you will use the manual more as a user manual than as a reference. This will allow you to understand how the various parts of the program relate with one another. Once you become more familiar with the program, you will probably use the reference chapter more often for finding quick answers to your questions. ARexx script writers will especially find this reference chapter a necessity.

1.5.1 Documentation Conventions

While reading through this manual, you will see four different types of text styles—plain, **bold**, **typewriter**, and *italics*. These styles are used to denote certain information.

Most of the manual is in the plain style, which denotes regular text with no particular emphasis over other material.

The **bold** style is used to denote labels (text) used in the GUI (Graphical User Interface), such as the **Execute** button on ADPro's main window.

The **typewriter** style denotes something that you, as the user, would type on the Shell or in one of the input fields on the GUI.

Text written in the **typewriter** style can also mean that it is something you can type from the keyboard. Some examples include **F10**, **Ctrl**, **Shift**, and **Del**.

If a particular equivalent requires two keys be held down, it will be listed as such: **Shift + F1** (with a space between the two keys). Going from left to right, you would hold down each key and press the final (rightmost) key. In the previous example, **Shift + F1** would mean that you should hold down the **Shift** key (either one) and, while still holding down the first key, press the **F1** key.

The *italics* style is reserved, as well, for information that you need to type in. The difference is that information in this style must be defined by you. For instance, in Section 5.1.1 when it asks for *ADPro-directory*, the manual cannot put a directory name in its place because it doesn't know where you actually installed the ADPro program.

The *italics* style is also used to describe variables in mathematical formulae.

1.5.2 Example Images

Several of the miscellaneous example pictures used in the User Manual and for the tutorials are based on images from the PhotoDisc™ collection (Volume I: Business and Industry and Volume II: People and Lifestyles CD-ROMs). They have been scaled, cropped, reduced in color, and converted to an appropriate image format so that they would fit on a floppy disk and be usable with ADPro. These images are only included on the distribution disks for educational (tutorial) purposes and **must not** be redistributed.

1.6 Acknowledgements

ASDG has been assisted by its beta testers and those who have sent in manual corrections and program suggestions. To all of you, we extend our appreciation. We would also like to thank the CDTV community as well as the CDTV staff at Commodore for helping to drive our technology forward by coming up with so many suggestions.

We would also like to thank those of you who have supported our company's efforts by not pirating our software.

1.7 About ASDG

ASDG Incorporated has been a leader in technological innovation and quality since the Amiga premiered. The company has created a number of firsts in the Amiga market, including the first recoverable ram disk, memory boards for the A2000, floppy accelerator, IEEE-488 interface, expansion serial board, and the first 24 bit-plane color image processing system.

In addition, the company has been honored by being the first inductee into Commodore Magazine's "the Amiga Public Domain Hall of Fame," and the company president, Perry Kivolowitz, was given the first Amiga Community Service Award by Amazing Computing magazine.

In recent years we have refined our focus to be the input, processing, and output of color images. In fact, our ADPro package has become a standard tool among Amiga owners who work with pictures. ASDG was even recognized by Commodore as having made a significant contribution to the development of CDTV (as a consequence of our development of color imaging tools). For every year that the award has been given, Art Department Professional has been honored with Amazing Computing's Reader's Choice Award for Best Image Processing Program.

In addition to image processing software, we also distribute CygnusEd Professional, a winner of a Reader's Choice Award for Best Text Editor (an honor it has also won every year the award has been given out), and T-Rexx Professional, a highly-praised and powerful set of applications for the Toaster user and video professional.

We're always looking for quality products which will enhance our product line. Please feel free to contact us concerning our possible publishing or marketing of your color imaging related product. Our mailing address and telephone number are listed in Appendix A.

Chapter 2

Installation

This chapter describes what is required to install and run the Art Department Professional programs. This information is **very important**, so be sure to understand it completely.

2.1 System Requirements

The Art Department Professional package is compatible with the entire family of Amiga computers. These include the A500, A600, A1000, A1200, A2000, A2500, A3000, and A4000 Amiga computers. All programs in the ADPro package **require** at least AmigaOS 2.04 to run; they are also compatible with AmigaOS 2.1 and 3.0. For A500, A600, A1000, and A2000 machines, an accelerator is highly recommended. A hard drive is also highly recommended.

2.2 Memory Requirements

By their very nature, the Art Department Professional programs require a lot of memory. In fact, ADPro can utilize as much contiguous memory as you have on your machine; memory is contiguous if it's in a single chunk. While it can run in as little as one megabyte of memory, we suggest a minimum of four megabytes of contiguous expansion memory in order to get any substantial work accomplished. Also, if you use the “high quality” features found in many parts of the program, more memory may be required.

If you use the **SETCPU** program, you should specify the **HEAD** option to allow

your 32-bit memory to merge properly with your 16-bit memory. On machines with only 32-bit memory, the **HEAD** option is not required, but specifying it will not cause any problems.

Remember that fast memory must be **contiguous** for ADPro to make full use of it. The more contiguous fast memory you have, the larger the bit-maps (images) you will be able to process.

Ignoring the differences between the file formats that ADPro can process, ADPro really loads only two types of images: color and grayscale.

ADPro defines a grayscale image to be any image in which every color in the image's palette is a shade of gray (i.e., the red, green, and blue contents of each color are equal). ADPro considers anything which is not a grayscale image under this definition to be a color image.

2.2.1 Calculating Memory Requirements

To calculate how much memory would be required for a given color image, you can use the following formula as a guide:

$$MinimumBytesNeeded_{color} = Width_{pixels} * Height_{pixels} * 4$$

This guide will tell you the approximate minimum amount of *contiguous* fast memory (in bytes) required to process a color image of a given size. The multiplication by 4 to give the number of bytes required is actually a short hand for $\frac{32}{8}$. The 32 is the number of bit-planes that ADPro uses for all of its processing. The division by 8 turns the number of bits into bytes (8 bits in a byte).

Take as an example an image measuring 640 by 400 pixels in size. This would require a minimum of 1,024,000 bytes of contiguous available memory to be able to take advantage of all of ADPro's processing capabilities. Of this space, three quarters will be used for 24 bit-plane raw image data. The remaining quarter is used to hold rendered display data.

ADPro converts all color images into 24 bit-plane data when the image is loaded. Grayscale image data is converted into 8 bit-planes rather than 24. This produces a memory requirement rule-of-thumb as follows:

$$MinimumBytesNeeded_{gray} = Width_{pixels} * Height_{pixels} * 2$$

The same 640 by 400 pixel image (as in the previous example) would require a minimum of 512,000 bytes of contiguous memory if it were a grayscale image rather than color.

For more information about ADPro's memory requirements, and for a greater understanding of how ADPro internally stores image data, please refer to Section 3.2.1.

2.2.2 Limiting Memory Usage

On systems with a great deal of memory, you can limit the amount of **fast** memory which ADPro will allocate. By default, ADPro will allocate approximately 131,072 bytes (128K) less than the largest fast memory block available. If you wish ADPro to use less memory than this default, you can specify it in two ways:

1. From the Workbench, you can set a Tool Type (called **MAXMEM**) to the maximum number of bytes that ADPro will be allowed to use for its primary image buffer. For example, specifying **MAXMEM=3000000** will limit ADPro's primary image buffer to 3,000,000 bytes, even if you have 8 megabytes of free contiguous fast memory available.

Tool Types can be set from the Workbench using the **Information...** command (under the **Icons** menu). For more information on how this is done, please consult your Amiga system software documentation.

2. From the Shell, you may specify a command line option of **MAXMEM=bytes**. For example, specifying the command **ADPro MAXMEM=3000000** will have the same effect as in the Tool Type example.

The amount of fast memory used will be slightly larger than the amount specified as a memory limit. This extra space is used to hold the program's code in memory, as well as some small data structures.

For more information about Tool Types and Shell options, please consult Section 13.1.

2.2.3 Reducing Memory Usage

ADPro's enhanced palette support consumes as much as 300,000 bytes of memory. If you know that you will not make use of this feature, you can disable it and reclaim the memory it would have used.

From the Shell command line, including the **NOENHANCED** (or the shorter **NE**) option will disable enhanced palette operation.

From the Workbench, you can specify a Tool Type called **NOENHANCED**. If present, enhanced palette operations are disabled. Otherwise, they are enabled by default.

2.3 Installation

This section gives an overview of the installation process, while the subsections describe key points for each installation mode.

Insert the ADPro Program Disk 1 into a floppy disk drive. The name of this disk is “ADPro_D1”.

Double-click on the ADPro_D1 disk icon. After the disk's window opens, double-click on the **Install-ADPro** icon. This will start the program that will help you install ADPro onto your hard disk.

The Installer utility will then execute the installation script. Follow the directions given in the installation to place the ADPro and FRED programs, as well as their support files, where you want them.

After the Installer has successfully completed the installation, remove the ADPro disk from the floppy drive and store it in a safe place. If your floppy or hard disk somehow becomes unreadable in the future, you will be assured of having an original working copy of the program available to you.

If you own the Professional Conversion Pack (PCP) or any of our scanner, film recorder, or other expansion module packages, please perform the installation of the module patches **after** installing the ADPro package. These patches can be found on the disk named, “ADPro_D4”.

If you own DPaint AGA version 4.5 from Electronic Arts, please perform the appropriate patch (based on your copy of DPaint AGA) also **after** installing the ADPro package. These patches, found on the disk named, “ADPro_D6”, are required to properly use the DPAINT loader and saver modules.

NOTE Due to a problem with the Installer program, be sure that none of the following assignments are deferred (using the **defer** keyword): **ADPRO:**, **ADP_FRED:**, **ADPROSCRIPTS:**, **FREDSCRIPTS:**, **REXX:**, **S:**, **LIBS:**. If they are, be sure to assign them (i.e., **Assign ADPRO:**) before beginning the installation.

2.3.1 Novice User Installation

The “Novice User” option is disabled because the installation of ADPro requires that you specify the location of its many files. As these can take up a few megabytes of hard drive space, it is up to you to select an adequate location for them.

2.3.2 Intermediate User Installation

You should select the “Intermediate User” option if you want to be asked the minimum number of questions to install the files.

Although you have control over the location of each installed program, we suggest you create a new drawer for ADPro and place its related programs, utilities, and scripts in the same drawer (in appropriately named subdirectories).

For example, if you had a partition named “**Work**” and it had enough space to hold the files, you might want to create a new directory called “**ADPro**” using the **Make New Drawer...** command on the installation program panel that asks for the drawer to place the ADPro program. All the related files could be placed in this same directory, or in subdirectories of it.

NOTE If you will use the Installer program’s function to create a new directory (i.e., **Make New Drawer...**), it is highly recommended that you **DO NOT** include spaces in the drawer name. Following this advice will avoid any problems accessing the directory, especially from ARexx scripts.

Note that some directory assignments will be added to your **S:user-startup** file. If they already exist, they will be updated.

NOTE If you already own MorphPlus, you should probably install ADPro in the same directory so that the modules from both programs can be shared from the same user interface.

Consult the ReadMe file on the first distribution disk for tips on doing this.

2.3.3 Expert User Installation

You should select the “Expert User” option if you want the most control over which files will be installed and in what directories on your system. Be prepared to answer many questions.

You should read the previous section (“Intermediate User” installation) for general installation tips.

As you have more control over the location of files, we suggest you place all the scripts and utilities in subdirectories of the ADPro directory, such as:

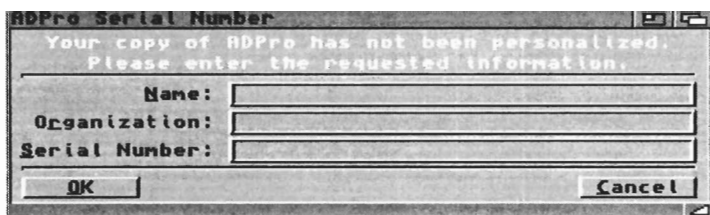


Figure 2.1: The ADPro Serialization panel.

Program	Suggested Directory Name
ADPro	<i>partition:ADPro</i>
FRED	<i>partition:ADPro</i>
ADProScripts	<i>partition:ADPro/ADProScripts</i>
Miscellaneous ARexx Scripts	<i>partition:ADPro/ADProScripts</i>
FREDScripts	<i>partition:ADPro/FREDScripts</i>
Utilities	<i>partition:ADPro/Utilities</i>

where *partition* is one of your partitions, such as “Work” or “Workbench”.

Of course, if you already have a directory reserved for graphics-related applications, you might want to create the “ADPro” directory within it.

Note that you will be asked whether some directory assignments should be added to your **S:user-startup** file. If you decide not to add them, these assignments will still be made for the current session (i.e., they will be valid until you turn off or reboot your computer), just not placed in your **S:user-startup** file.

2.4 Serialization

The ADPro program needs to be serialized before it can be used. You can serialize it during the installation process, or do it afterwards. The first time ADPro is run, it will display the serialization requester (as shown in Figure 2.1).

The Name and Organization are optional, but the Serial Number **must** be entered. Enter each item in the fields provided. To move to the next field, press the **Tab** key; to move to the previous field, press the **Shift + Tab** key sequence. Your ADPro package should’ve included a serial number. Enter it into this field as printed.

Once you’ve entered the serial number, pressing the **Return** key while in the Serial Number field (or selecting the **OK** button) will accept the current settings and attempt to serialize the program. If there is a problem serializing the

program, a message will be displayed. Otherwise, the program was serialized properly, and the Name, Organization, and Serial Number can be viewed from within the program by displaying the About panel (select the **About...** menu item under the **Project** menu).

After successfully personalizing the program, the release notes for the program will be displayed. This is the same file that is displayed if you selected the **Release Notes...** menu item from the **Project** menu of the program.

Chapter 3

Basic Concepts

Before you can take advantage of ADPro's many features, you should learn some basic concepts such as knowing how to use the program's controls, loading, saving, and displaying images, and understanding how your images are represented in memory.

This chapter will take you through the simplest steps involved in understanding these concepts. You will then build upon these concepts in later chapters.

3.1 Navigating the User Interfaces

This section gives general information about various parts of the ADPro's user interfaces. Basic terms such as screens, panels, buttons, and input fields will be described. The use of the file requester, chooser, and meter window will also be discussed.

3.1.1 User Interface Objects

Throughout this manual you will see references to various user interface objects, such as control screens, control panels, buttons, gadgets, input fields, cycle gadgets, rocker switches, and sliders. This subsection will give definitions of each of these, along with their use. The use of keyboard keys will also be discussed.

You should use this as a reference for all user interface terms described in this manual. If a term used in this manual has a different meaning than what will

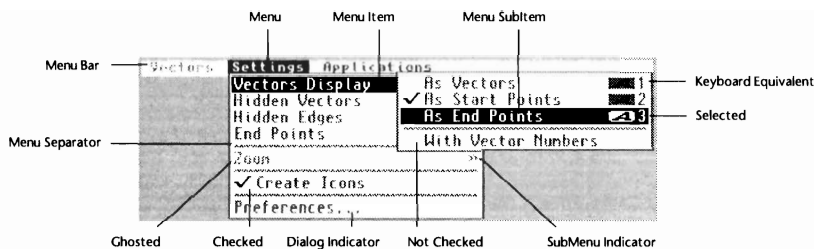


Figure 3.1: An example menu strip hierarchy.

be described below, it will be noted to you.

Control Screens

A control screen (or screen for short) is a large area of the Amiga display. In ADPro, they are used to display the main controls and, for some operators, operator-specific controls. All of the screens have front-to-back gadgets, which allow you to flip between screens. Some have drag bars, which allow you to move your screen up and down to reveal other screens behind it.

Some screens have a menu bar attached to them. You can check if a screen has a menu bar by depressing the right mouse button (menu button). The menu items and subitems can be browsed by holding down the right mouse button, moving the mouse pointer up to the menu bar's (top of the screen) menus, and moving down through the menu items and subitems.

The menu bar has the following hierarchy:

```

menu
  menu item
    menu subitem

```

and a sample of this is shown in Figure 3.1.

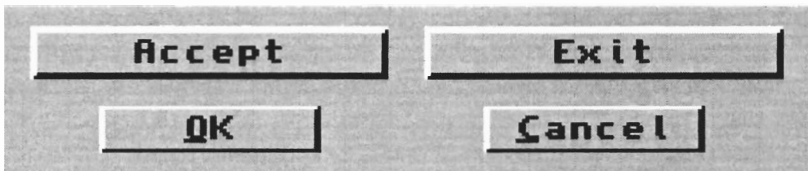
Menus are the phrases displayed on the menu bar. Menu items are the phrases displayed below each menu. Menu subitems are the ones displayed to the right of a particular menu item (ending with ">>").

To select a menu item or subitem, move the mouse pointer above the item or subitem and release the right mouse button.

Control Panels

A control panel is a smaller area of the Amiga display that can move around on a control screen. Some panels look like standard Amiga windows (as viewable on the Workbench screen), while others have a slightly different 3-dimensional look. Some of these panels can be shrunk to a smaller size and later expanded to full size.

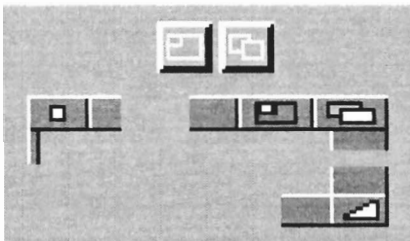
Buttons



Buttons are areas within windows that are outlined by a raised 3-dimensional border. Most of the time a short text string or group of strings is displayed within it. Also, whenever the left mouse button (selection button) is pressed while the mouse pointer is above this area, the button usually changes to a recessed look. In the manual, the terms “press”, “select”, and “click on” are used interchangeably to describe the selection of a button.

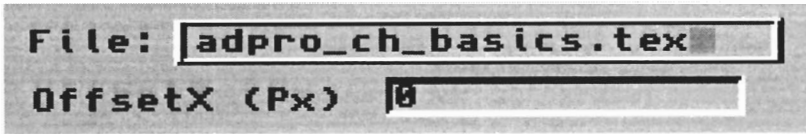
The term “double-click” is used to describe the action of selecting the button twice in rapid succession.

Gadgets



Gadgets are the same as buttons, although they describe the system’s buttons—close gadget, zoom gadget, front-to-back gadget, sizing gadget, etc.

Input Fields

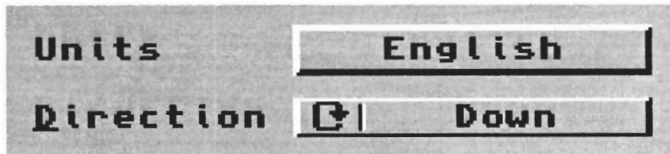


Input fields are areas within windows that either have a recessed 3-dimensional look or a ridged outline around them. They accept input from the user, such as a number or a text string.

NOTE After you finish entering a number or string in one of these input fields, you **must** press the **Return** key for it to be acknowledged by the program.

For some input fields, pressing the **Return** key will move the cursor from field to field. Other input fields require the **Tab** key to be pressed instead. For those types of fields, pressing **Shift + Tab** will move to the previous input field.

Cycle Gadgets and Rocker Switches

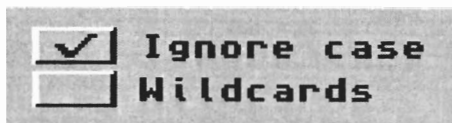


Cycle Gadgets and Rocker Switches are specific, though similar, types of buttons. When selected, they cycle through the possible choices available for that attribute.

The differences between a cycle gadget and a rocker switch are in the way they look and the way they behave. A cycle gadget uses the Workbench 2.04 or later's imagery—an arrow pointed in a circular, clockwise direction. To cycle forward through the choices, click once on the button. To cycle backward, hold down the **Shift** key while clicking on the button.

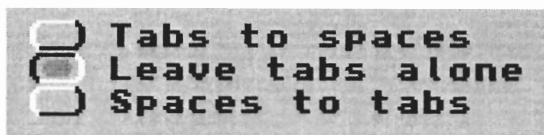
A rocker switch looks like a regular button, but it behaves differently depending upon which horizontal half of the button you click. If you click on the right side of the button, it will cycle forward through the available choices. If you click on the left side, it will cycle backwards.

Check Boxes



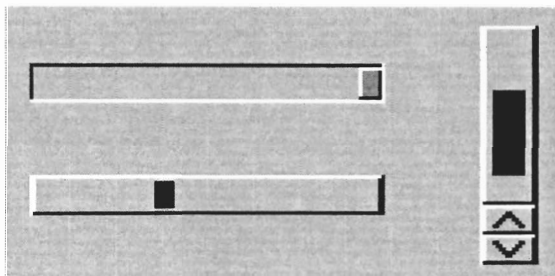
A check box is a button that represents a two-state (on or off, yes or no, etc.) setting. A check mark will appear on the button for the positive choice (on, yes, etc.), whereas an unchecked box signifies the negative choice (off, no, etc.). Click once on the button to toggle between positive and negative.

Radio Buttons



Radio buttons are a set of two or more oval shaped buttons where each button represents one choice among many choices. These choices are mutually exclusive (only one can be selected at a given time). They are called radio buttons because of their functional resemblance to buttons on a radio, where each button would represent a different radio station. The currently selected choice is displayed as a recessed button. Selecting a different item will recess that item's button and un-recess ("pop up") the previous button.

Sliders and Scrollers



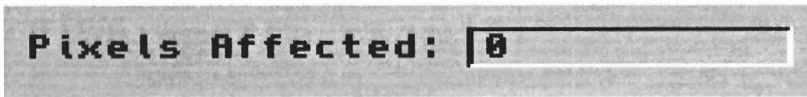
A slider is a gadget that lets you specify a value. A knob (rectangle within the slider area) describes the value in relation to the minimum and maximum values. For example, a slider that can specify a value between 0 and 100 will have its knob in the middle of the slider area for a value of 50. You can also use the left mouse button to select the knob within a slider and drag it to a new position. As you drag the knob, the value of the slider will be updated in

real-time in the integer input or text field located to the right of the slider.

A scroller looks similar to a slider, except that the knob describes the position or amount of viewable area within a larger area. For example, the standard Amiga windows have scroller bars (including scroll arrows) to display parts of a disk or drawer's contents. To view other parts of the area, you can drag the slider knob or use the scroll arrows to view the hidden areas in smaller increments.

Sliders and scrollers can either be oriented horizontally or vertically, although most of the time (at least in the ADPro programs) you will see them horizontally.

Text Fields and Display Boxes



A text field, or sometimes called a display box, is an area of a window or panel that contains a word or phrase that conveys some information to you but does not allow you to change it. Often, this field is outlined by a recessed box, signifying that the value (the text) cannot be modified.

Keyboard Equivalents

Some of the operators have keyboard equivalents for frequently used functions. Either one key can be pressed, or a series of keys can be pressed in a particular order. See Section 1.5.1 for a discussion of this.

Also, some control panels have “local” (only usable within the panel itself) keyboard equivalents. They are indicated by an underscore character (‘_’) underneath the keyboard equivalent. Instead of using the mouse to select, toggle, or activate a particular user interface object, you would simply press the keyboard equivalent.

3.1.2 The File Requester

Anytime you save or load a file, ADPro invokes its file requester. This subsection describes how to use the file requester to specify the name of a file to be loaded or saved.

The ADPro modules use one of two different types of file requesters. The main file requester is described first, followed by the second type.

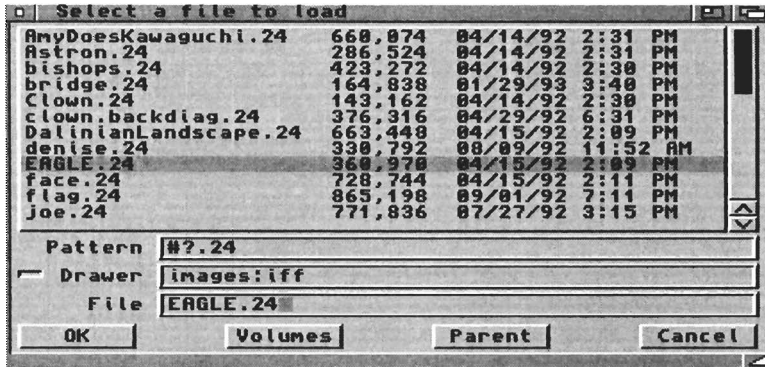


Figure 3.2: The ASL File Requester.

The ASL File Requester

A major part of the requester contains the list of files and subdirectories in the currently selected **Drawer**. You can select a file by clicking on its name. The selected filename will appear in the **File** input field (below this area).

Also displayed in this list area are all the subdirectories contained within the current directory. Clicking on these entries allows you to examine subdirectories. The subdirectory name will appear in the **Drawer** input field and the files within the subdirectory will appear in the list area.

If a directory contains more files and subdirectories than can be displayed in the list area, you can use the scroll bar and arrow buttons (to the right of the list area) to scroll through the current directory's contents.

To exit out of the current directory and back to its parent directory, select the **Parent** button.

Another way of selecting the directory (drawer) is to directly type it into the **Drawer** input field. Remember to press the **Return** key so that the file requester can acknowledge your selection.

If you want to see all of the available disks and devices, select the **Disks** button; in some versions of the operating system this button is labelled as **Volumes**. You can then click on one of the listed entries to examine it.

As an alternate to using the scroll bar and arrow buttons, you can use the cursor up and down keys to scroll the currently selected file.

The **Close** gadget and the **Cancel** button do the same thing—they exit from the file requester without selecting a file. You would do this if you decided not

to select a file at this time.

If you wanted to confirm your choice for a file, select the **OK** button. Alternatively, if the cursor is currently in the **File** input field, pressing the **Return** key will accomplish the same thing.

Filtering Files From View

The **Pattern** input field can accommodate a string conforming to a simple grammar. The grammar is sufficient to define nearly any combination of filenames you may want to display.

The best way to explain ADPro's use of AmigaDOS-style wildcards and pattern matching for the **Pattern** field is to plow right through the explanation and give examples. Here's the explanation:

The grammar includes wildcards defined in AmigaDOS format. This means that single character substitutions can be defined as "?" while multiple character substitutions can be defined as "#?".

Patterns may include the or-operator, "|". There is no need for an and-operator.

And here's some examples:

- To match all files not ending in ".o", you would set it to: `~(##?.o)` meaning "match anything that does not end in an ".o".
- To match all files ending in ".c", ".asm", ".h", ".i", or the file "Makefile", you would set the **Pattern** field to: `(##?.c|##?.asm|##?.h|##?.i|Makefile)`
- To match all files not ending in ".o" nor ".bak", you would set **Pattern** to: `~(##?.o|##?.bak)` (or, alternatively, `~(##?.(bak|o))`)

ADPro also permits nested patterns. So, for instance, the first example could be rewritten as: `(##?.(c|asm|h|i)|Makefile)`

The REQ File Requester

The REQ file requester, available through the `req.library` file in your `LIBS:`, is less-often used for loading and saving files.

A major part of the requester contains the list of files and subdirectories in the currently selected **Drawer**. You can select a file by clicking on its name. The selected filename will appear in the **File** input field (below this area).

Displayed in the right hand list area is the device list. The device list will contain an entry for each of the following types of entities:

1. Mounted disk volumes, such as **DFO:**, **RAM:**, or **VDO:**
2. Logical disk names (the name by which each physical disk device is "labelled")
3. Assigned names, such as **L:**, **FONTS:**, and **LIBS:**

Notice that the device list is always kept alphabetically sorted, and that the device list is automatically updated when disks are inserted or removed.

Clicking on an entry in this list changes the **Drawer** specification and displays that drawer's contents in the file list area. If the **Drawer** input field is blank, this indicates that the current directory is the directory containing the ADPro program.

If a directory contains more files and subdirectories than can be displayed in the list area, you can use the scroll bar and arrow buttons (to the right of the list area) to scroll through the current directory's contents.

As the current directory is being read, its contents will be listed in the order they are encountered. This order will almost certainly not be sorted. Should you wish the file list to be sorted alphabetically at any time, simply click the left mouse button on the slider to the right of the file list or upon the scroll arrows above or below the slider. This will instantly sort the contents of the file list. You can also accomplish this from the keyboard by depressing **Alt + Return** twice.

To rescan the current directory, you can press the **Get Dir** button.

To exit out of the current directory and back to its parent directory, select the **Parent** button.

Whenever a directory is currently being scanned, you can determine when it is done by watching the status line above the **Parent** button. While it is scanning, you should see an ellipsis (...) at the end of the line. When it is done, the word, **Done**, will replace the ellipsis.

This status line also shown how many files are shown (visible) and hidden. Files can be selectively shown and hidden by changing the text strings in the **Hide** and **Show** input fields. For example, if you enter a file pattern of ***.txt** in the **Show** field, only those files that end in **.txt** will be visible.

You can also show or hide any icon files (ending in a **.info**) by toggling the button above the assigned directory list (at the right side of the file requester).

Another way of selecting the directory (drawer) is to directly type it into the **Drawer** input field. Remember to press the **Return** key so that the file

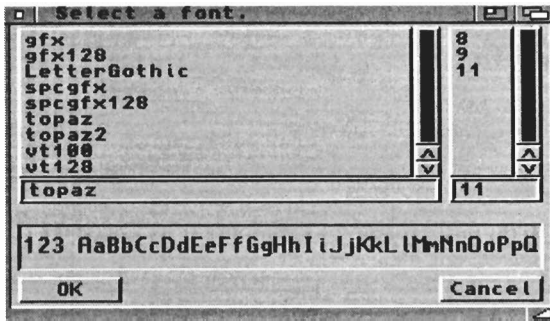


Figure 3.3: The ASL Font Requester.

requester can acknowledge your selection.

The **Close** gadget and the **Forget it** button do the same thing—they exit from the file requester without selecting a file. You would do this if you decided not to select a file at this time.

If you wanted to confirm your choice for a file, select the **OK** button. Alternatively, if the cursor is currently in the **File** input field, pressing the **Return** key will accomplish the same thing.

3.1.3 The Font Requester

The font requester contains two scrollable lists, the left one displays the fonts available in the current **FONTs:** directory and the right one displays the point sizes available for the currently selected font. To specify a different font directory, assign **FONTs:** to that directory. To specify more than one directory, use the **Assign FONTs: directory ADD** command from the Shell.

The font (left side) list will contain the name of each valid Amiga font located in the font requester's current directory. The font requester will show both fixed-width and variable-width (proportional) fonts. Clicking the left mouse button upon a font name will cause the defined sizes of the font to be displayed in the sizes (right side) list area. You may then click the left mouse button upon one of the displayed sizes to cause ADPro to attempt to change to that font and size.

3.1.4 The Screen Mode Requester

A Screen Mode requester allows you to describe the type and dimensions of a screen. The Screen Mode requester used in ADPro also adds the ability to set

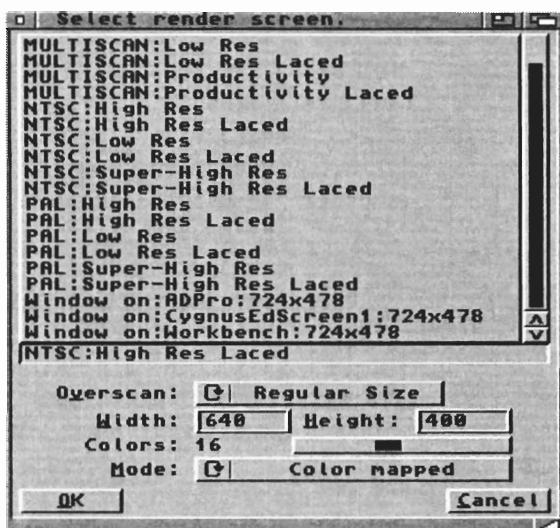


Figure 3.4: The Screen Mode requester used in ADPro.

the mode and number of colors. This custom screen mode requester functions similarly to the ASL, except for the additional controls and the absence of the Property List requester (which gives more information about a particular screen mode).

A major part of the requester is a list of available screen or render modes that your hardware provides. Simply select the mode by clicking on its entry in the list. If no display modes appear or some which you believe should be there are missing, be sure to check that the monitor icons are installed properly. See Section 9.1 for more information on installing monitor icons.

The **Overscan:** cycle gadget allows you to select the type of overscan to use for the currently selected screen mode. These modes are defined by the settings in the Overscan Preferences editor. The overscan setting affects the screen size and when it starts autoscrolling.

The **Width:** and **Height:** input fields allows you to customize the dimensions of the screen.

NOTE If you enter a size which is greater than the default for a particular mode, you will only see an area *default* pixels wide by *default* pixels tall at any given time. In these circumstances, you can use the screen auto-scroll feature to navigate around this large “virtual” screen.

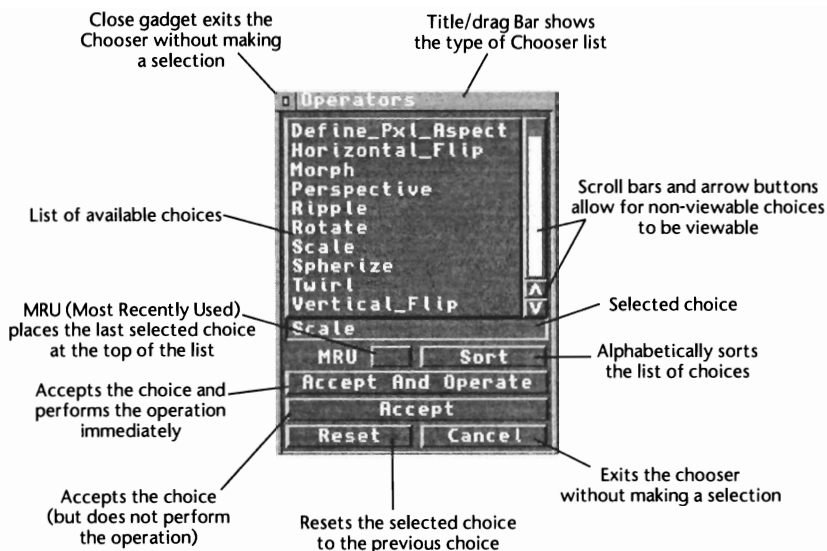


Figure 3.5: An example of a Chooser list.

The **Colors:** slider allows you to select the number of colors to use for the rendered image.

The **Mode:** cycle gadget allows you to select which color mode to use for the rendered image.

3.1.5 The Chooser

A special type of requester, called the Chooser, will be displayed during some operations. The Chooser allows you to “choose” one item from potentially many.

The chooser opens a window (shown in Figure 3.5) and presents all of the available choices in a scrollable list box. If there are more choices than will fit in the box, you can use the scroll bar or arrows to scroll through additional items in the list. Under AmigaOS 2.04 and 2.1., the bottommost item appears to be in a recessed box rather than an raised box like the rest of the list. This item is not actually part of the scrollable list, but rather displays which item in the list is currently selected. Under AmigaOS 3.0 and later, this recessed box does not exist. Rather, the currently selected item is always highlighted.

You can double-click upon any item in the list. If there is a button in the chooser window marked **Accept And Something** (for example, **Accept And**

Save), then double-clicking on an item in the list will both accept that item and start whatever action is appropriate. If the **Accept And Something** button does not appear in the chooser, then double-clicking on an item will simply accept that item but cause no further action.

For Chooser lists with many entries it can become somewhat time-consuming having to scroll through the items just to get to a particular one. To help navigate through the items in the list, you can press any key from A through Z to jump to the first list entry that begins with that letter. That entry will become the selected item. If no entries exist for a given letter the list will not move at all.

The items in the list are kept in alphabetical order by default. However, this is not always the case. The checkbox marked **MRU** can exist in two states, either checked or not. **MRU** stands for “Most Recently Used”. When checked, the item you selected and accepted the last time you saw this list will be moved from its alphabetical place and placed at the top of the list. This is especially useful when you are toggling back and forth between two or three choices in the list. Using the **MRU** feature, these two or three items will be found right next to each other at the top of the list. This makes using the chooser even faster.

Next to the **MRU** check box is a button marked **Sort**. Selecting this button causes the list to become alphabetically sorted if it had gotten out of order due to the **MRU** feature.

There will always be an **Accept** button in the chooser. When there is no **Accept And Something** button, the **Accept** button is the same as double-clicking on an item in the list. When there is an **Accept And Something** button in the chooser, the **Accept** button causes the selected item to be accepted but no further action taken.

The **Reset** button causes the item which was selected when you last entered the chooser to become active again. The **Cancel** button exits the chooser without making any choice. This is the same as selecting the close gadget at the top left of the chooser window.

3.1.6 The Meter Window

Whenever ADPro is processing your image, a meter window is displayed showing the amount (percentage) of completeness of a particular operation.

As you can see in Figure 3.6, the meter is a “level” indicator extending from the left to right. The description of the operation currently being performed (or of the percentage of completeness) is displayed within the meter itself.

There is also a **Stop** button to interrupt the operation.

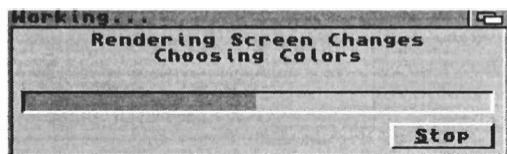


Figure 3.6: The meter window showing an operation 50% complete.

3.1.7 The Visual User Interface

There are a few ADPro modules that use a WYSIWYG interface. WYSIWYG stands for “what you see is what you get”. WYSIWYG user interfaces are characterized by a very strong visual component (hence the name “visual user interface”) with immediate feedback given to the user. The different modules within ADPro which use the visual interface have common features and operating characteristics. This subsection describes these attributes.

Figure 3.7 shows the control screen for the Crop_Visual operator, which is typical of the operators that use the visual interface.

First you will note that operators which employ the VUI open their own screen. Contrast this to the non-VUI operators, such as Scale, which open a small panel.

The VUI wouldn't be very visual if an image didn't play a prominent role in the screen. In Figure 3.7, you'll note that the entire left hand portion of the screen is used to display an image of the area being worked upon. This image is displayed in 8 shades of gray and is dithered to enhance the quality of the image. We chose to convert the image “on the fly” to grayscale, since we have such a small number of colors to work with on a standard Amiga display. Being usable on the lowest common Amiga configuration was a primary design goal in developing the VUI.

Note that as the image is being drawn, you can generally interrupt the rendering either by clicking on the button (or selecting the menu item, if one is available) marked **Stop** or **Abort** (or, if the visual interface supports it, hitting **Shift + a**).

ADPro will always scale the image so that either the height or the width will fill the screen. This allows you to see as much of the image as is possible, as large as possible. Since the Amiga's display is wider than it is tall, images with a horizontal aspect will “fill” the screen better than images with a vertical aspect. In Figure 3.7, you'll note that the image fills the height of the screen but does not fill the width. This is because the portion of the image on-screen has an overall vertical aspect.

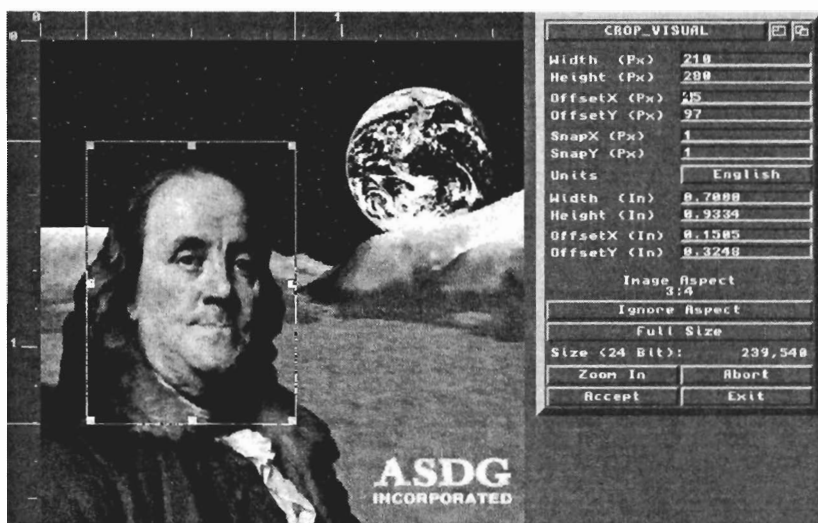


Figure 3.7: The Crop_Visual control screen, a typical user of the Visual User Interface.

Note that the VUI takes into account the current settings for X and Y resolution. For example, if the X resolution were set to 100 DPI and the Y resolution were set to 300 DPI, a bit-map 100 by 100 pixels in size would be drawn scaled as a stretched out rectangle. While the IFF format supplies an X and Y resolution to ADPro as the image is loaded, you may need to set these values yourself occasionally. See Section 8.12 for more information about setting the X and Y resolution for a given picture.

You can specify the type of VUI screen with the **Screen Mode...** command (under the **Settings, Set Visual** submenu) and **Deep** toggle subitem. The **Screen Mode...** command displays a screen mode requester, from which the screen mode for all VUIs can be selected. The **Deep** toggle specifies whether it will open as a 3 bit-plane (8 shade) or 8 bit-plane (256 shade) screen; the 8 bit-plane option may not be available if your graphics hardware doesn't support that screen depth.

Along the top and left borders of the screen you might find a set of rulers. These rulers demark inches or centimeters, depending upon which unit of measure you have selected. These measures, though, are dependent upon ADPro's knowledge of the resolution of the image. The rulers do not break down into fine detail. In inches, for example, tick marks are drawn only down to the $\frac{1}{8}$ inch level. More accurate measuring information is usually given in the module's control panel.

To complete our tour of the typical VUI control screen, let's move to the top of the control panel. The button-like area at the top, in this case labeled **CROP_VISUAL**, is the drag bar for the window. By depressing the left mouse button in this area and holding it down, you can drag the entire window around the screen. You can't drag it off the screen, however. Some modules may use a different style of panel. Specifically, the panel looks like a normal window like those found on your Workbench screen. The drag bar for these windows is also at the top of the panel.

Most of the VUI modules' control panels have a "zip" gadget. You can shrink the panel to a smaller size so that you can see more of the image. When the control panel is small (or zipped), you can still manipulate the objects on the image with your mouse. Some modules implement keyboard shortcuts for various commands which can be used while the control window is zipped.

Finally, there is the front/back gadget for the screen. Use this as you would any other screen front/back gadget.

3.2 Understanding the Image Data Flow

Understanding how image data is represented and used by ADPro can be very beneficial for knowing what ADPro is doing with your images. This section will describe how your images are stored, as well as how they "fit" within the image data flow.

3.2.1 Internal Data Types

Within ADPro, image data can take several forms. Understanding the differences between these data types will help you understand how and why ADPro works the way it does.

The first distinction is that image data can be one of two types: *raw* or *rendered*. Rendered image data is data which contains color mapped information. Images such as those which are displayable on the Amiga's screen are examples of rendered image data. For example, an IFF file which contains a 16 color image and can be displayed with any viewing program contains a color map describing the actual color that will be displayed for each value in the image. Without the color map information, the image data can be unrecognizable.

ADPro can create, read, process, or write rendered data with up to 256 color mapped colors. A 256 color rendered image contains 8 bit-planes. Therefore, ADPro can create, read, process, or write rendered data with up to 8 bit-planes.

Raw image data is divided into color and grayscale types. Common to both

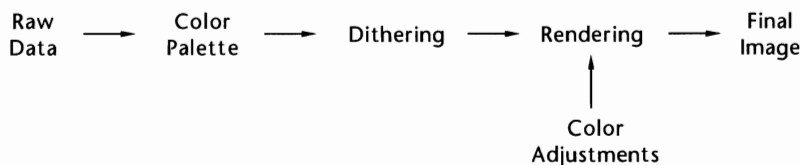


Figure 3.8: Data flow within ADPro.

types of raw image data is that a color map is not required in order to make the image data recognizable. Grayscale raw image data is stored internally in ADPro in 8 bit-planes (24 bit-planes, or 16.7 million shades for color).

Memory permitting, ADPro always converts rendered image data into raw image data during a load operation. The color map of a rendered image file is examined. If it contains only shades of gray, the rendered image data is converted into an 8 bit-plane raw gray image. If any non-gray colors are detected, the image is converted into a 24 bit-plane raw color image.

The **Execute** button causes ADPro to process raw image data into rendered image data. Figure 3.8 shows how ADPro will process the raw image data into rendered image data.

Nearly all of the functions which ADPro can perform require the presence of raw image data. Only one of color or gray raw image data can be processed at one time. Operators are provided to convert between the internal formats of color and gray raw image data.

Some functions in ADPro require only one type of raw image data but not the other. ADPro will display a message if the right type of data is not currently available.

ADPro will display (in the **Image Information** area, as shown in Figure 3.9) the types of image data available. If only rendered image data is available, the word **Rendered** will appear. If only raw color image data is available, the word **Color** will appear. If only grayscale image data is available, the word **Gray** will appear. If both raw and rendered image data is available, both Raw (**Color** or **Gray**) and **Rendered** will be indicated.

ADPro also tells you the width and height of the currently defined image, as well as how much (as a percentage) of the total primary image buffer is occupied by the image. The capacity of the temporary buffer is also displayed. Note that the primary image buffer percentage includes the percentage used by the temporary buffer.

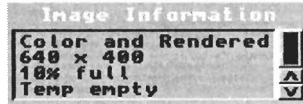


Figure 3.9: The Image Information area, showing the existence of raw and rendered color image data for a 640 x 400 image.

3.2.2 ADPro's Image Buffers

As described earlier, ADPro has one image buffer which it allocates on start-up. This buffer, however, doesn't just store one type of data. In fact, it is this buffer that holds the raw image data, rendered image data, and temporary buffer data.

The raw image data buffer (also known as "raw space") contains the 8-bit (for grayscale) or 24-bit (for color) image data for the image currently loaded.

The rendered image data buffer (also known as "rendered space") contains the Amiga-displayable image data—the representation of the image in ADPro's image buffer which can be directly displayable on an Amiga without additional hardware. Of course, your standard Amiga graphics hardware may not support the display of all possible Amiga-displayable images (i.e., AGA images).

The temporary (or "temp") buffer contains the raw 8- or 24-bit image data which was last saved by an invocation of the TEMP saver. This image data is separate from the primary buffer's image data, although the contents of both buffers might be the same.

The cumulative size of all three "buffers" is limited by the size of ADPro's image buffer, which in turn is limited by either the largest contiguous block of memory available at the time you started ADPro or the value specified with the **MAXMEM** tool type or command line option, whichever is smaller.

A special type of buffer, called the alpha buffer, is used to store alpha channel image data loaded using the **As Alpha** load type option.

NOTE Alpha channel information is stored in the same space as the rendered image (i.e., render space). So, if you are working with images that are only rendered data (i.e., the Image Information area says that it's "Rendered"), they will be overwritten by the alpha channel image data unless you invoke the **Rendered.To.Raw** operator on it.

Loading an image using the **As Alpha** option is not identified in the Image Information area at all, even with the word "Rendered". This is to identify that no rendered data representing the raw image data exists. Otherwise, if

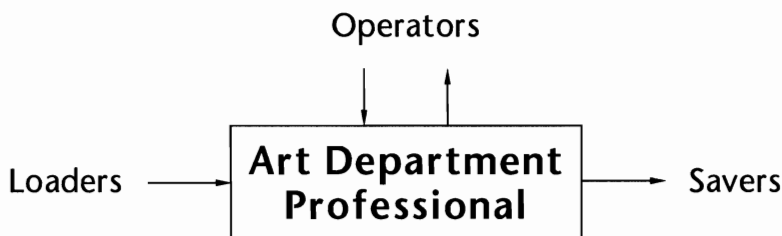


Figure 3.10: ADPro's Building Blocks.

"Rendered" were displayed after an image was loaded into the "alpha buffer", you might assume that the rendered image represents the raw image...which it doesn't.

3.2.3 ADPro's Building Blocks

The beauty of ADPro is that much of the program is modular, made up of separate parts that add additional functionality to the main program. These modules are called Loaders, Savers, and Operators. Figure 3.10 indicates how these modules interact within ADPro.

A **loader** is a source of image data, typically from a file on disk, but can be from a scanner, digitizer, frame grabber, tape drive, or other external device which can produce image data.

A **saver** is a destination of image data, also typically a file on disk, but can be a graphics board, film recorder, printer, tape drive, or other external device which can accept image data.

An **operator** is a manipulator or analyzer of the image data in ADPro's image buffer. Operators are extremely flexible since they allow input, processing, and output of either raw or rendered image data.

3.3 Understanding How Loaders Work

Loaders do not necessarily retrieve image data from a disk file—each file to be loaded from disk is searched for a color map. If a color map is contained in the file, it will be loaded along with the image data, provided that the ADPro palette is not in a **Locked** state. If the palette is **Locked**, ADPro will ultimately ask you if you really do wish to load the palette from the file.

If found, the palette will be inspected to see if it contains only shades of gray. If it does, the file's data will be expanded out into an 8 bit-plane grayscale image. If not, the file will be expanded out into a 24 bit-plane color image.

Depending upon the state of the **Orientation** setting, the image may be rotated 270 degrees while it is being read from disk.

Depending upon the **Composition** setting, the image may be composited relative to the current image in ADPro's image buffer.

The expansion into 8 or 24 bit-planes is usually performed after the file has been loaded. However, if you are loading an Amiga displayable file and there isn't enough memory to hold the raw image data but there is enough memory to hold the displayable data, then it will be loaded without the conversion to 8 or 24 bit-planes. In this case, the image can be displayed, but none of the processing commands which depend on having raw data around will function.

When loading files which are already in an Amiga displayable format, the image is directly viewable immediately after loading by selecting the **Redisplay** button.

Depending on the load format, aborting a load while in progress may or may not produce displayable image data.

3.3.1 What a Loader Does Internally

This section describes one of the most significant steps that loaders perform when reading in an image. If the file being loaded contains rendered image data (as opposed to true color or raw image data), that data will be converted into 24 bit-plane color raw image data inside ADPro (if the image being read is rendered grayscale, it will be converted into 8 bit raw grayscale inside ADPro).

Note that even a tiny 1 bit-plane image containing, for example, only red and green, will be converted into 24 bit-planes of raw data. That is, *the space required to process an image can be many times larger than the space needed by the image while it is on disk.*

If there is not enough memory to convert a rendered image back into raw image data, or if you abort the conversion, no raw data will be available for you to manipulate within ADPro. Many of ADPro's advanced image processing functions require the presence of raw image data in order to operate.

However, there are some important functions you can still perform even when there is no raw image data available (either because you didn't allow it to be created or because it wouldn't fit into available memory).

If the rendered image data you just loaded is displayable by the Amiga, you

can view the image directly by hitting the **ReDisplay** button. Also, you can immediately save the image out into a format which supports the same type of rendered data (i.e., IFF-ILBM or IFF-ANIM format).

From ARexx, you can force rendered image data to *not be converted* into raw image data by using the **NOPAD** loader option.

You can also force rendered image data to *not be created* by using the **ONLYPAD** loader option.

These two capabilities will be mentioned in Section 13.8.5.

3.4 Understanding How Savers Work

Depending on the save format, there might be a restriction on the type of image data (raw or rendered) that can be saved. This is based on what is currently available in ADPro, as well as what the saver will allow. If the type of data is not available, an error message will be displayed.

For many savers which save to files on disk, if only raw image data is available, a file requester will instantly appear for the specification of a filename to save the raw data. This behavior is different than in previous versions of ADPro, which would show all data types options in a control panel but only let you select the raw image data choice. Now, you will only be given a choice if more than one form of the data exists.

3.5 Understanding How Operators Work

Once the image has been loaded into ADPro, various manipulations can be performed on it. These manipulations come in the form of operators. The ADPro package comes with several operators, all of which are described in Chapter 8.

As can be seen in Figure 3.10, operators have complete freedom to read, process, and rewrite raw image data maintained within ADPro.

NOTE It is important to note that unlike changes to the raw or rendered data made by any of the screen mode or color controls, changes made by operators to raw image data cannot be undone without reloading the raw image data.

Also, operators are not, in general, fully interruptible. Since operators actually modify the raw data maintained by ADPro, interrupting an operator in mid-

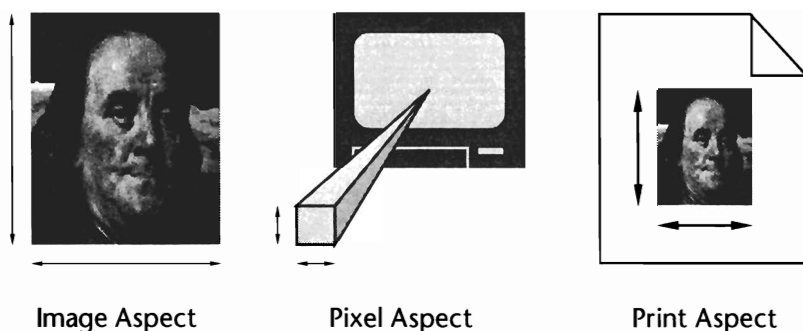


Figure 3.11: Comparison of various aspect ratios.

operation will leave partially operated-upon data in memory or may, in some cases, cause all data to be lost (if interrupted).

3.6 Understanding Aspect Ratios

This section will explain and clarify two types of aspect ratios which you might have heard about before—image aspect and pixel aspect.

An aspect ratio is the relationship between the width of an item to its height, expressed in the form:

$$\text{width} : \text{height}$$

For example, a 1 : 1 aspect ratio means that the item (whatever is being measured) is as wide as it is high.

Aspect ratios are described by saying their width value to their height value. Taking the previous example, you would say that it has a “one-to-one” aspect ratio.

Image aspect refers to the aspect ratio of the image itself. The *width* and *height* correspond to the width and height (both measured in pixels) of the image, as shown in the leftmost image in Figure 3.11. For example, a 640 x 400 image has an image aspect ratio of 640 : 400 (or more simply, 8 : 5).

Pixel aspect refers to the aspect ratio of an individual pixel on your display device (i.e., your Amiga monitor). The *width* and *height* correspond to the width

and height of a single pixel, as visualized in the middle image in Figure 3.11. This ratio is different for each display device, and can also be different for similar display devices if their horizontal and vertical adjustments are different.

3.7 Exchanging Data

Exchanging image data with other computers is divided into two steps:

1. The ability to read and write image files in the logical format used on other computers.

The ADPro (and Professional Conversion Pack) loader and saver modules solve this requirement by allowing you to read and write files logically structured in the BMP, PCX, and JPEG formats, to name a few. The companion (but separate) product called the Professional Conversion Pack (PCP) includes the TIFF (miscellaneous platforms, specifically Apple and IBM), TARGA (principally IBM), and the Rendition (miscellaneous platforms) formats.

2. Physical connectivity, or how to get the information physically from one computer type to another.

This second problem can be solved in a number of ways and is not addressed by the ADPro package. One low-cost solution that works quite well for exchanging data with IBM PC type computers is a product called CrossDOS from Consultron ((313) 459 - 7271). It will allow IBM 720K 3.5 inch disks to be used on the Amiga as if they had been native Amiga disks. Also, Workbench 2.1 and later include the ability to read from and write to these same 720K disks. Used in conjunction with these products, ADPro can directly read and write IBM 720K diskettes. IBM 1.44MB diskettes are also supported with high-density floppy drives.

Other solutions to the connectivity issue, based upon network connections, are slow in emerging but making steady progress. At the very least, a serial port based connection made via modem or null-modem will do the job.



Chapter 4

Tutorials

This chapter contains several lessons for you to follow. Each of these lessons focuses on a different part of the Art Department Professional package. After successfully completing these lessons, you should be able to use the Art Department Professional applications more effectively.

The following lessons are included in this manual:

- Lesson 1. ADPro: File Format Translation
- Lesson 2. ADPro: Color and Display Adjustments
- Lesson 3. ADPro: Using the Palette Editor
- Lesson 4. ADPro: Image Compositing
- Lesson 5. FRED: Batch Image Translation
- Lesson 6. FRED: Batch Image Processing
- Lesson 7. FRED: Creating an ANIM From a Sequence of Images
- Lesson 8. FRED: Creating a Fade Between Two Images
- Lesson 9. Sentry: Automated JPEG Archiving

These lessons assume that you have a basic knowledge of loading and saving images and selecting and executing operators. If you are unfamiliar with these concepts, please read Sections 5.3 (loading images), 5.4 (saving images), and 5.5 (executing operators).

4.1 Lesson 1. ADPro: File Format Translation

Level: Beginner/Intermediate

Many people purchase ADPro to transform their files from one format to another. They look to ADPro to bridge the gap between applications, taking their Impulse files and converting them into IFF24s, or Framestores into JPEGs. Whatever the reason, ADPro performs this most basic of operations easily and quickly.

This tutorial will take a 24-bit IFF image and convert it into a JPEG file. The IFF file used in this tutorial are available in the Tutorials directory that was created during installation. If you selected not to install the tutorial files, please do so now before continuing.

4.1.1 Setting Up

First, let's start the ADPro program:

1. Move to the Workbench screen, if it isn't the frontmost screen.
If another screen is currently visible, use the **Left-Amiga + M** combination (hold down the **Left-Amiga** key, then repeatedly tap the "M" key to cycle through all open screens until the Workbench screen is visible).
2. Locate the ADPro icon (it is a picture of Ben Franklin, with the name "ADPro" underneath it). ADPro should be on one of your hard disk partitions (or a drawer within it), so double-click on the icon for the hard disk which contains the ADPro program.
If ADPro wasn't installed in the root directory of this partition (you would see its icon in the partition's window), keep double-clicking on drawers until you find it.
3. Once you've found the icon, simply double-click it to start the program.
After a few seconds, the main ADPro window should appear.

4.1.2 Selecting the Loader

To load a file that is in a particular image format, we first need to specify the loader module to use. Loaders are small programs which handle the conversion between an external image format (such as Impulse, Framestore, JPEG, etc.) into that of ADPro's internal image format. Many loaders come with ADPro.

For this tutorial, we need to select the IFF loader.

1. From ADPro's main window, press the "L" key to open the floating Loaders list. Move the window off of ADPro's window so that you can see the controls.
2. Scroll this list until you see the loader that corresponds to the format of the image you're trying to load. We will be loading an IFF file, so search for the "IFF" entry in the list.
3. Double-click on the IFF loader to run it. (Double-clicking on an entry in the **Loaders** list or any of the other lists automatically selects and runs the module.)

4.1.3 Loading the File

Most file format loaders ask you to specify the name of the file to load from a standard file requester.

1. Change the directory (**Drawer**) to the location of the IFF file you want to load. Either type it in (remembering to press the **Return** key after entering it) or using the mouse and **Parent**, **Disks** or **Volumes** buttons to navigate to the directory.

For this tutorial, we will want the image named "**NightScene.24**" located in the directory where you installed the tutorial files.

2. Choose the file to load by clicking on it, then selecting the **OK** button to initiate the load. Alternatively, you can double-click on the filename to do both operations.

The loading process will take anywhere between instantaneous to a few seconds, depending upon the dimensions of the image. A meter window will appear showing a graphical representation of the progress.

At this point, the image data is now loaded in ADPro's image buffer.

Notice how the **Image Information** area shows "**Color**" and "**768 x 337**". This shows that there is raw color (24-bit) image data in ADPro's buffer, 768 pixels wide by 337 pixels high.

You can now perform any operation on it or save it, which is what we want to do.

4.1.4 Selecting the Saver

Before we save a file, we must first select the saver module. Savers, like modules, are small programs, but they handle the conversion between ADPro's image data format and an external image format.

In this tutorial, we need to select the JPEG saver.

1. From ADPro's main window, open the floating Savers list by pressing the "S" key. Move this list away from the main window.
2. Scroll the **Savers** list until you see the JPEG saver.
3. Double-click on the JPEG saver to display its control panel.

4.1.5 Saving the File

Each saver will have its own parameters which must be set. These parameters are shown within a control panel. For the JPEG saver, you can control how much compression (which affects the quality) is done to the image.

1. For JPEGs, we usually want this set to 32. Change this to 32 if it isn't there already.
2. Select the **OK** button to accept these settings.
3. A file requester will appear for you to specify the directory and name to give the JPEG file which will be saved. Enter this information, using a ".jpeg" filename extension to identify this file on disk as being in JPEG format.

Using filename extensions describing the format of an image is usually a good thing.

4. Select the file requester's **OK** button to start the save process.
A meter window will appear showing the progress of the operation. Once it has been saved, control will return to the ADPro main window.

Congratulations! You have just performed a simple IFF-to-JPEG file conversion.

4.1.6 Using the UNIVERSAL Loader

What if you had to convert a few files, but each one was in a different image format? There maybe modules for each format, but it would be cumbersome to switch the loader module for each file.

ADPro comes with a special loader called "UNIVERSAL". By its name, it can handle pretty much any image file you throw at it. The UNIVERSAL loader handles the format detection and running of the proper module to actually load it. Think of this loader as a smart front-end to the other file format loaders.

1. From ADPro's main window, double-click on the UNIVERSAL loader. Note that you could've double-clicked the loader from the floating Loaders list—these **Loaders** buttons are another way of launching loaders. A file requester will appear, asking for a file to load. Let's select the JPEG file that we just saved to see how UNIVERSAL can automatically detect the file format.
2. Change the **Drawer** and **File** to the previously-saved JPEG file.
3. Double-click on the JPEG filename to automatically load it. Almost instantly, the JPEG loader's control panel appears asking you to specify how to load the JPEG file.
4. Select the **OK** button to accept the default values and watch the JPEG file load into ADPro.

4.1.7 Conclusion

In this tutorial, you have learned:

- How to start the ADPro program.
- How to select and run a loader module.
- How to select and run a saver module.
- How the UNIVERSAL loader can save time and hassle.

4.2 Lesson 2. ADPro: Color and Display Adjustments

Level: Intermediate

You have an image loaded in ADPro's image buffer, but how do we display it? This tutorial will go through some of the basics of color and display adjustment.

An understanding of loading images into ADPro is necessary for this tutorial.

4.2.1 Setting Up

To start off, load the tutorial file named "**Plane.256**", located in the tutorial directory. Use the **UNIVERSAL** loader for convenience.

4.2.2 Specifying the Amiga Render Screen

The first thing we'll do is set up to display our image to a native Amiga screen mode.

1. Select the **Set Render Screen...** button to display the available display modes.
2. Scroll the list until you see an NTSC or PAL Low Res mode, then click on it once.
3. Drag the **Colors:** slider knob to the **32** setting. This is a good setting which will work on all Amiga models.
4. Select the **OK** button to accept these choices.

In the Set Render Screen requester, you can also specify overscan resolutions. You can also type in custom screen widths and heights, which affect the viewing area and when, for images larger than the screen size, autoscrolling is enabled.

You might want to try these after completing this tutorial.

4.2.3 Rendering and Viewing the Image

Now that we've specified the resolution of the screen and the number of colors, let's render our image to the screen.

1. Select the **Execute** button.

This operation examines the raw image data and creates rendered image data in the number of colors you specified in the Set Render Screen requester.

As the image is rendered, you will see it appear on screen in a top-down manner. If the image is taller than the screen it's being displayed on, you will see it start redrawing at the top of the screen. This is normal, and is an indication that the image is taller than the render screen.

Once it's finished drawing, the top part of the image will appear to "snap" back into view.

2. Use the cursor keys or numeric keypad keys to scroll the image. By using the **Alt**, **Shift**, and **Ctrl** keys in combination with the cursor keys, you can adjust the amount of scrolling that will be performed.
3. Once you are done, click the left mouse button. This will return you to ADPro's main window.
4. To redisplay the render screen (once it has been opened), simply select the **Redisplay** button.
5. To toggle between the render screen and ADPro's main window, repeatedly press the **Spacebar**.

4.2.4 Using the Dither Modes

You might've noticed that the image looked very coarse or "chunky" in 32 colors. To increase the apparent number of colors, we'll incorporate some dithering.

1. Toggle back to the main window, if not already there.
2. Using the right mouse button, move to the **Display** menu, down to the **Dither** menu item, and over to the **Floyd** subitem. This will place the menu checkmark next to **Floyd**, meaning that the next rendering will use a Floyd-Steinberg dither.
3. Select the **Execute** button again to display the raw image data using the Floyd dither.

Note that you cannot simply press the **Redisplay** button because all that it will do is show the last rendered image, not the image that would be rendered with the Floyd dither.

4. Click on the image when done viewing it.

4.2.5 Adjusting the Balancing

Next, we will adjust the color balancing of the image. The rendered image data gets passed through the balancing settings before being displayed. By

adjusting some settings you can alter the appearance of your image *without affecting the raw image data*.

1. Open the Balancing control panel by selecting the **Balancing...** menu item under the **Display** menu.

Notice that you can control the red, green, and blue components independently, as well as the brightness, contrast, and gamma levels.

For this tutorial, we will increase the gamma level to lighten the image without causing the brighter parts of the image to be “washed out” to white, as would the brightness control.

2. Set the **Gamma:** level to 15 and press the **OK** button.
3. Select the **Execute** button to recreate rendered image data, this time filtered through a gamma of 15.

Notice how the image is brighter, but you can still see the detail in the image. Let’s compare this to an equivalent increase in brightness.

1. Go back into the Balancing control panel and set the **Gamma:** level back to 0.
2. Now, switch the **Brightness:** level to 15, select **OK**, and rerender (i.e., select the **Execute** button).

Now notice how the image is lighter, but what was previously light-colored is now white or near-white. If you are seeking an increase in brightness but don’t want any loss in detail, use the gamma control.

You should go back and reset all the controls back to 0, then rerender, before proceeding.

4.2.6 Adjusting the Palette

Now, let’s see how changes to the palette affect the rendered image data, as well as which colors are used.

1. Open the Palette control panel by selecting **Palette...** from the **Display** menu.
2. Select the **Edit...** button to display the 256 color palette editor.
This screen contains 16 rows by 16 columns of color registers. Color register 0 is the upper-left square or “color well”.
3. Click once on register 0. Notice how the **R**, **G**, and **B** sliders are updated to show the register’s value.

4. Change the color for this register to **R** = 0, **G** = 0, **B** = 255, which is pure blue. If you are changing these values by entering them into the input fields, be sure to press the **Return** key after entering each number. Change register 1 (the square to the right of the current one) to 0, 255, 0 (pure green).
Change register 2 to 255, 0, 0 (pure red).
What we have done is set these registers to custom colors.
5. Select the **OK** button from the palette editor, then the **OK** button from the Palette control panel.
6. Select the **Redisplay** button.

Notice that parts of the image are now red, green, and blue. This is because the **Redisplay** button displays the current rendered image data using the current palette contents (which were just modified).

What do you think would happen if we pressed the **Execute** button instead? Try it out and compare the results.

1. First, click the left mouse button to return to the main window.
2. Select the **Execute** button.

Notice how the image has returned to the previous colors. This is because the **Execute** button recalculates the best possible palette based on the raw image data, whereas **Redisplay** displayed the image with the current palette.

This is an important distinction. Understanding this difference in operation will avoid unexpected results when working with palette adjustments.

Switch back to the main window by pressing the **Spacebar**.

4.2.7 Displaying to a Window

So far we've been rendering to a separate screen. Let's try out an alternate display "destination"—rendering to a window. ADPro allows you to select a public screen on which to open up a render window. Instead of displaying it on a separate screen, it will appear in a window that you can move around, resize, and pan.

1. Select the **Set Render Screen...** button.
2. All the window options (i.e., public screens currently open) are available at the bottom of the list, so scroll down to it. You should see an entry beginning "Window on:Workbench". Double-click on this entry to select it.

3. Now, press the **Execute** button.

Notice that instead of a separate screen opening, the image is rendered into a window on the Workbench screen. This window has the standard drag bar, scroll bars within the window's borders, and a sizing gadget. Play around with these controls. You can pan around within the window by using the window's scroll bars, the cursor keys on the keyboard, the keypad keys, and the mouse (to drag the image around).

Of course, the number of colors used is limited to the number of colors in your Workbench screen, as well as the Workbench's palette. Rendering to a window will not always give you an exact representation of the image, especially since it is limited by the screen's palette. What it does offer you is a way of seeing your image in a smaller screen area, as well as giving a quick "preview" of the image.

For operations which require exact examination of the image, your best choices will still be rendering out to a 24-bit graphics board or to at least a separate screen with the maximum number of colors supported by your graphics hardware. For example, if you had an AGA-equipped machine, you could set your Workbench to 256 colors and render to a window on it for a more realistic representation of your image.

4.2.8 Conclusion

In this tutorial, you have learned:

- how to view an image and pan around in it if it larger than the visible area
- how to toggle between the render screen or window and the main ADPro window
- how to use the dither, balancing, and palette controls
- how the above controls affect the raw and rendered image data
- how to display an image on a separate screen, as well as in a window on any public screen, including the Workbench screen

4.3 Lesson 3. ADPro: Using the Palette Editor

Level: Intermediate

ADPro has many powerful palette controls. This tutorial is aimed at walking you through the flexible color range functions available. Both explicit and implicit ranges are discussed.

4.3.1 Setting Up

We first need to load in an image and create rendered image data.

1. Load in any color image.
2. Convert it to grayscale using the `Color_To_Gray` operator.
3. Be sure that the **Available Modes Only** menu item (under the **Settings** menu) is **not** checked. This will allow us to render in 256 colors on any Amiga model.
4. Select the **Set Render Screen...** button (from the main control window).
5. Change the **Colors:** control to **256**.
6. Select the **OK** button to accept this choice.
7. Select the **Execute** button to create rendered data.
On non-AGA machines you will not be able to view this rendered data because your graphics hardware doesn't support it. That is fine.
On AGA machines, a 256 color representation of your image will appear. Just press the left mouse button to return to the ADPro window.
8. Select **Palette...** menu item under the **Display** menu to open the Palette control panel.
9. Select the **Edit...** button to display the Palette Editor.

4.3.2 Explicit Ranges

Color register 0 will be selected yellow.

1. Select the **SELECT RANGE** button. Notice that it stays highlighted.
2. Select color register 255 (bottom right corner). Notice all colors in between are highlighted in green.

3. Select color register 0 again (top left corner). Change this color to white by setting the **R**, **G**, and **B** sliders to 255.
4. Select **COPY RANGE**. Notice it stays highlighted.
5. Select color register 255 (bottom right corner). Notice all colors in-between have changed to white.

Summary: When an explicit range is defined, all members of the explicit range which are green are affected by a range command.

4.3.3 Colors Not In A Range

1. Choose a color register along the left edge of the color register grid, somewhere in the middle (of the grid's height).
2. Select the **DESEL RANGE** button. Notice it stays highlighted.
3. Select the color register on the same line as the previously selected register, but along the right edge. Notice all of the registers in this line are no longer highlighted. They have been dropped from the explicit range.
4. Select color register 0 (top left corner). Change it to pure blue (R,G,B = 0,0,255).
5. Select the **COPY RANGE** button. Notice it stays highlighted.
6. Select color register 255 (bottom right corner). Notice all colors in-between have changed to blue except for those not in the range, which were unaffected.

Summary: When an explicit range is defined, colors which are not members of the range (even if they are in the middle of the range) are not affected by range commands.

4.3.4 Range Commands and Explicit Ranges

1. Select the **DESEL RANGE** button. Notice it stays highlighted.
2. Select color register 0. All green highlighting should now be gone. You should have a color grid with all pure blue except for one line of pure white.
3. Select the first white color.
4. Select the **SELECT RANGE** button.
5. Select the last white color. Notice all of the white colors are now highlighted with green.
6. Select color register 0 (top left corner). Make this color pure red.
7. Select the **COPY RANGE** button.

8. Select color register 255 (bottom left). Notice that only the colors in the explicit range were affected.

Summary: When an explicit range is defined and a portion of the range is contained between the end points of a range command, only those colors in the range are affected by range commands.

4.3.5 Range Commands and Implicit Ranges

1. Select the first highlighted color.
2. Select the **DESEL RANGE** button.
3. Select the last highlighted color. No color should be highlighted with green at this point.
4. Select color register 0 (upper left corner) which should be pure red.
5. Select the **COPY RANGE** button.
6. Select the color register still containing blue immediately before the first register containing red. Notice all color registers in between turn red.

Summary: When no explicit range is defined, all range commands work on an implicit range defined by the selected color before the command button is hit...and the color selected after the command button is hit.

4.3.6 Place Holders

If you select a color register or range of color registers with either **Alt** key held down, the register or range of registers will be highlighted with red instead of green.

Colors highlighted in red are part of the **explicit** range but their color values are locked. They will contribute a “place holder” to range commands, but their colors will not be affected by range commands.

This is best explained with the **RGB RANGE** and **HSV RANGE** commands and another example:

1. Ensure that no color registers are currently highlighted with red or green (i.e., no explicit range defined).
2. Select color register 0. If it isn't already, make it pure red.
3. Select color register 15 (top line, right corner). Make it pure blue.
4. Select the **RGB RANGE** button. Notice it stays selected.

5. Select color register 0. Notice that color registers 0 to 15 now contain a ramp from red to blue done in 16 steps.
6. Set colors 4 through 11 to pure white.
7. Define an explicit range of colors 0 to 15 by selecting 0, hitting the **SELECT RANGE** button, and selecting color 15.
8. Select color 4. Hit the **SELECT RANGE** button. Hold down the **Alt** key and select color 11. You should now have an explicit range defined from 0 to 15 with colors 4 through 11 being highlighted in red and the others highlighted in green.
9. Select color register 0. Hit **RGB RANGE**. Select color 15. *You should see nothing change.* This is because you computed a red to blue ramp with 16 steps, same as before...and it didn't affect the colors highlighted in red.
10. Select color register 4 (the first white color). Hit the **DESEL RANGE** button. Select color register 11 (the last white color). You should be left with 8 green highlighted colors separated into two groups of four, with 8 highlighted colors inbetween.
11. Select color register 0. Hit **RGB RANGE**. Select color register 15. Notice that the 8 white colors are unchanged but the 8 highlighted colors will change. They still contain a ramp from red to blue but the ramp is now done in 8 steps, not 16.

Summary: Having colors in an explicit range highlighted with red means they contribute a place holder to range commands, but are not affected by them.

4.3.7 Conclusion

In this tutorial, you have learned:

- How to create explicit and implicit ranges.
- How to copy one color to a range of color registers.
- How to protect certain color registers from being affected by range operations.
- How to use color registers as "place holders" for range operations.

4.4 Lesson 4. ADPro: Image Compositing

Level: Intermediate

Many people use ADPro for its image compositing features. This tutorial is intended to give you some exposure to the basic image compositing tools available. If you have never used ADPro for compositing, we hope that you will find more uses for it after going through this tutorial. If you have composited images before, we hope this tutorial teaches you some useful tips.

In this tutorial, we will create a composite of a city skyline during a lightning storm. Topics discussed will be compositing with a transparent color, as well as using an external alpha channel. This should give you a good exercise in image compositing.

4.4.1 Setting Up

All of the images we will need are in the directory where you installed the tutorial files, so all we need to do is start up the program. Do this now.

4.4.2 Loading the Background Image

The first step is to load in the background image (i.e., the first layer in our multi-layer compositing exercise). When you are doing compositing, you start with the lowest “layer” or image, and composite each layer above it one image at a time.

1. Before we load the image, first make sure that we are not in Compose mode. Do this by selecting the **Replace** menu item under the **Loaders** menu.
Replace mode means that the next image to be loaded will replace the existing image in ADPro’s image buffer.
2. Select the **UNIVERSAL** loader to display the file requester.
3. Load the file called “**Lightning.256**”. Since the lightning will be behind our skyline, we need to load it first.

4.4.3 Compositing Using a Transparent Color

Our next step is to compose the skyline atop the lightning. One problem, however, is that we want the sky in the sky in the image to be transparent. Let’s use ADPro’s transparent color capability to set this color in the incoming

(skyline) image as transparent. This will allow the lightning to appear where the sky currently exists.

1. Switch to Compose mode by selecting **Compose** from the **Loaders** menu.
2. Use the **UNIVERSAL** loader to load the file called “**NightScene.24**”. This is our skyline scene.

Once loaded, the Compositing screen and control panel will appear. This is where you can specify the offset of the foreground image relative to the background image, how much of the foreground image to mix with the background, and a transparent color.

1. So you can get a better idea of the operation we’re performing, render the current image in ADPro by selecting the **Render** button. A grayscale representation of the lightning image will appear on-screen.
2. The incoming skyline image is represented as a box outline which you can position with the mouse or by entering values into the offset fields. Alternatively, you can use the cursor keys to “nudge” the box around in smaller increments. For this tutorial, drag the box representing the incoming image so that the bottom of it is aligned with the bottom of the background image; the widths of both images are the same, so the X offset should be 0.
3. Ensure the **Alpha:** cycle gadget is set to **Transparent Color**.
4. The **From R:**, **G:**, and **B:** sliders control the transparent color. Specify blue (R = 0, G = 0, B = 255) as transparent.
5. We want the total effect of this image, so make sure the **Mix:** slider is set to 100.
6. Select the **Compose** button at the bottom of the panel to compose the skyline above the lightning.

Note that using transparent color works best if the transparent color only appears in the area that you are trying to “key out”. Otherwise, what can happen is that areas outside of the intended area (which are the same transparent color) will become transparent as well—not what you intended. This problem can appear if you specify black as the transparent color, but parts of your image which you didn’t want transparent are also black. As black is a common color, this can be an issue. For our tutorial, our skyline was specially created on a blue background (somewhat to mimick the traditional blue and green screen process).

Although you might want to see the composited image, let’s wait until we get back into the Compositing control screen. We can render the image in there.

4.4.4 Compositing Using an Alpha Channel (Matte)

The final “layer” will be “**Plane.256**”. However, it is on another backdrop. How can we only use a specific part of an image? With an alpha channel.

Think of an alpha channel (or matte, as it’s also called) as a “hole-cutter” or external grayscale file where the luminance (dark to light levels) represent percentages of the foreground image which will be composited. In other words, if your alpha channel image was a white square on a black background, if your foreground was the image of a face, you’d only see a square cut-out of the face.

We will compose our plane image onto our night skyline scene. Our matte (“**Plane.matte**”) will cut the plane out of the sky.

1. Compositing using an alpha channel requires two files: the image to composite and the image to use as the matte. We first need to set up the program for compositing. Since we are already in **Compose** mode, we can continue with the next step.
2. We first need to load the alpha channel into “alpha memory”, a special place in ADPro’s buffer. Select the “**Plane.matte**” file as the matte to load.
3. Our foreground and alpha images are the same width and height—this is a requirement. Let’s move the alpha image down and to the right, say 50 pixels for the X and Y offset each.

NOTE For alpha channel compositing, we need to position the alpha and foreground images at the same X and Y offset. This is another requirement, so remember these offsets for the next steps.

4. Select the **As Alpha** button to load the matte into “alpha memory”.
5. Now choose the foreground image, named “**Plane.256**”.
6. From the Compositing control panel, we need to position the foreground to match our previously loaded alpha. Change the offset values to 50, 50.
7. We now need to specify that we want to use the image in “alpha memory” for compositing. Switch the **Alpha:** cycle gadget to **Alpha Memory**.
8. Select the **Compose** button to compose our foreground using the alpha channel previously loaded.

4.4.5 Conclusion

In this tutorial, you have learned:

- how to use transparent color and alpha channels to composite images
- how to use the cursor keys to “nudge” the placement of your composited images
- what steps are required for alpha channel compositing
- how to mix the incoming image with the background image

We hope you can find some use of these techniques in your own work.

4.5 Lesson 5. FRED: Batch Image Translation

Level: Beginner/Intermediate

The main use of ADPro has traditionally been file format conversion. ADPro is great for that, but what if you have several files to convert? You certainly don't want to do them one by one...unless you have time to waste "babysitting" a program. This is something better left to a program that will do the same actions you would go through. That program is FRED.

In this tutorial, we will take several image files and convert them all from their current format to the same format (JPEG, for instance). Let's begin.

4.5.1 Setting Up

FRED and ADPro (or MorphPlus, if you're using it) are separate programs that work in conjunction with each other. FRED is the front-end that handles the task of sending the images and instructions to ADPro/MorphPlus (the back-end graphics "engine").

Both programs need to be running at the same time. To do this, follow the steps below:

1. Start FRED from its icon on the Workbench screen.
A 16 color high resolution screen should appear; if you're running FRED on an AGA-equipped machine, it will can open in 256 colors. Notice that there is hardly anything on it. This is because all of the controls are available via the menu bar.
2. Use the right mouse button to display the available choices in the menu bar.

Notice that the leftmost menu is **Project**. FRED works with projects, more commonly called "sequences". These sequences are the listing of image files which will be processed.

What we need to do now is start up our graphics engine.

1. Flip back to the Workbench screen by either clicking on the FRED screen's front/back gadget or holding down the **Left-Amiga** key and pressing the "M" key. This key sequence, when repeatedly pressed, will cycle through all of the open screens in your system at the time. Stop at the Workbench screen.

2. Find the ADPro or MorphPlus icon and double-click on it.
If the program was able to start up, you should see the ADPro/MorphPlus main interface.
3. We want to return to FRED's screen, so cycle back to FRED's screen with the front/back gadget or key sequence (as described above).
4. Use the left mouse button to click on the FRED screen. This activates the screen, allowing you to get at its controls.

Note that although you need either ADPro or MorphPlus (but not both) running at the same time as FRED to process images, you don't need to run it at the same time you start FRED—you can start anytime before you initiate the processing.

4.5.2 Creating a New Sequence

FRED handles images with sequences. A sequence is a collection of images. FRED represents sequences as a window on its screen, where the images are represented by “cells” in a horizontally scrollable view. These “cells” are actually references to the images (i.e., directions telling FRED where these images are located and what they are named), and NOT the images themselves. In fact, a sequence file is just an ASCII text file listing each images in a sequence window, in left to right order.

To create a new sequence, we need to open a new sequence window. Do this by selecting **New...** from the **Project** menu. A window should appear with a horizontal scroll bar (and arrows) and a title bar with “[] Untitled0.seq” in it.

The “[]” is the change indicator. It will contain an asterisk (i.e., “[*]”) once the sequence has been modified, either by something added to or removed from it.

The “Untitled0.seq” title is the default name given to this sequence file. If you save it to disk, it will have this filename—unless you modify it. All new sequence windows have a similar name, with just the number just before the “.seq” filename extension incrementing by one.

All sequence files, to be interpreted properly by FRED and other FRED sequence-compatible programs, should have a “.seq” filename extension on them. That is why FRED automatically places one on the default sequence name.

You are now ready to insert images into this blank sequence window.

4.5.3 Inserting Images Into a Sequence

You place images into the sequence window by inserting them. You can insert one file at a time, multiple files at a time, or a numeric range of files. We will show you the first two types of inserting at this point.

To insert a single image, do the following:

1. Select **Image(s)...** from the **Insert** submenu under the **Edit** menu.
This displays the Insert Images file requester.
2. Change the **Drawer** to the directory where your images are located.
3. Select the image file you want to add to the sequence. This can be done by clicking once on the name of the file (notice how it appears in the **File** field) and selecting the **OK** button.

Alternatively, you can double-click on the filename to do the same operation. This will probably be the way you will want to do this action in the future.

Each image that is selected is represented in the sequence window as a cell, a rectangle which initially displays a black square. This square area is where a stamp-size representation of the image would be displayed, if a stamp were made for it. This tutorial doesn't cover stamp creation.

Well, that inserted one image. But, we need to process several images at the same time (this is the meaning of batch processing, right?). Let's add more images to the sequence window, but this time adding them at the same time.

1. Open the Insert Images file requester the same way you did previously.
Notice how the **Drawer** field retained your previous setting.
2. Hold down the **Shift** key.

This tells the file requester that you are about to select more than one file. If you didn't have the **Shift** key held down and started clicking on files, each new selection would deselect the previous one—not what we want.

3. Start clicking on the files you want to add to the sequence.
Notice how each file is highlighted.
4. Release the **Shift** key and select the **OK** button to add all the highlighted files to the *end* of the current images in your sequence.

If you wanted to insert the images *before* a particular cell, you need to click on the cell to insert before, then select the files (as described above).

4.5.4 Saving the Sequence

Before we continue, we should save our sequence of images.

1. Select **Save As...** from the **Project** menu.
2. You need to enter a directory name in the **Drawer** field and name for the sequence in the **File** field.

For this tutorial, specify the **Sequences** directory that was created when you installed FRED. If you don't know where it is located, check the **ADP_FRED:** or **ADPRO:** directory. Otherwise, press the **Disks** or **Volumes** button to find it.

In the **File** field, enter **SampleImages.seq**. Be sure to append a **".seq"** extension to the filename. This type of extension is required for FRED and other FRED-compatible applications to recognize a FRED sequence.

When working with FRED, you should probably save your sequences often, especially before you begin processing a sequence with many image files. You don't want to reselect the files if something happens before or during the processing.

4.5.5 Selecting the Images to Process

FRED allows you to process all or a few images in a given session. This is done by first selecting the cells to process. In this tutorial, we will first select a few, then select all the images in the sequence to give you a feel for this feature.

1. Click once on the leftmost cell in the sequence to select it.
Notice now the previously raised border around the text below the stamp area has been lowered or "pushed in". This represents a "selected" cell.
2. Now, click on the cell to the right of this first cell.
Notice now the first cell's border "popped out" and the second cell's "pushed in". Only one cell can be "selected" at a given time.
But how do we select more than one cell to process? Use the **Ctrl** key.
3. Hold down the **Ctrl** key and start clicking on every other cell in the sequence, moving the horizontal slider to display the other cells in the sequence.

Notice how the text background has turned from gray to green. These cells are "highlighted".

NOTE More than one cell can be "highlighted," but only one can be "selected" (pushed in) at a given time. Also, this "selected" cell does not necessarily fall within the "highlighted" set of cells. FRED will process only "highlighted" and "selected" cells.

Now, if we were to process the sequence at this time, we would only get every other image. Most of the time you want to process all the images in a sequence, but be aware that you have the flexibility to work on any subset of images in a given session.

To highlight all the frames in the sequence, select the **Select All** menu item under the **Edit** menu.

4.5.6 Choosing the File Format to Use

Our next step is to specify which file format to use when saving the images.

1. Open the Invoke ADPro control panel by selecting **Process...** from the **Scripts** menu.

Note that this menu item can only be selected if you have at least one cell selected.

2. Select the **Add...** button and switch to the **FREDSSCRIPTS:FREDSavers** drawer.

Let's convert all of our images to JPEG—an operation which you might perform if you were to archive your images to save space on disk.

3. Select the **SaveAsJPEG.fred** script, then select the **OK** button from the file requester.

This one script will save each frame to disk in JPEG format.

4.5.7 Starting the Image Translation

Now we're ready to process the files.

1. Select the **Process** button from the Invoke ADPro control panel.
2. The ADPro screen will appear, asking you the quality level. You can just leave it at the default of 32, unless you want to use another setting. Consult your ADPro documentation for more information on this setting.

A list of options for modifying the filename will appear. Since we are saving to a different format, we probably would want the filenames to represent it in some way. Usually we want the filename extension to be appended or replaced with that of the output file format.

1. Double-click on the **"Append ???"** option from the list.
2. Answer **".jpeg"** when it asks for the string to append to each filename. This should be the default value, so you can just press the **OK** button.

3. You then be asked whether the saved JPEG files should be put in the same directory as the original or in a different directory. Select either one. If you select the latter, then you will be asked to specify an alternate directory.
4. Next, the **SaveAsJPEG.fred** script will ask if you want to create a FRED sequence file from the saved JPEG files. This is useful if you want to process the saved files within FRED or another program which can use FRED sequence files. Select the **No** option.

After these questions are answered, processing will begin. Each file selected is loaded into ADPro and immediately saved in JPEG format, with a “.jpeg” filename extension.

Processing will end once the Invoke ADPro control panel closes.

4.5.8 Verifying That Images Were Saved Properly

What you might want to do to confirm that they were saved in JPEG format is to flip back to ADPro’s screen, set the loader to JPEG, and do a test load of a few of the processed files. Remember, they should have “.jpeg” at the end of their filenames.

4.5.9 Conclusion

In this tutorial you have learned how to:

- create and save a new sequence
- insert images into a sequence window
- select images that will be used in the processing
- choose the image file format to convert to
- verify that images were actually saved to disk

4.6 Lesson 6. FRED: Batch Image Processing

Level: Intermediate

This tutorial will build upon what we learned in the previous tutorial (Batch Image Translation) by incorporating some image processing functions.

If you haven't completed that tutorial, please do so now before continuing with this tutorial.

4.6.1 Setting Up

If you are continuing from the previous tutorial, then go to the next section.

If you are restarting FRED after exiting it, select the **Open...** command under **Project** and choose the **SampleImages.seq** file saved in the previous tutorial.

4.6.2 Rearranging the Order of Processing

Let's play around with sequence windows and the order in which images are listed in them.

1. Open a new sequence window by selecting **New...** from the **Project** menu. Move the new window down so you can see both sequence windows.
2. Click the mouse pointer back in the **SampleImages.seq** window.
3. Select the **Select All** menu item under the **Edit** menu.
4. Select the **Copy** menu item under the **Edit** menu. Alternatively, you can use the standard key sequence, **Right-Alt + C**.
5. Click the mouse pointer in the new sequence window to activate it.
6. Select the **Reverse Paste** menu item under the **Edit** menu.
Notice how all the images in the first sequence have been inserted in the second sequence in reverse order.
7. Select the **Paste** menu item, also under **Edit**.
Notice how you have just created a sequence of images beginning and ending with the same image. This use of **Paste** and **Reverse Paste** is helpful in creating ping-pong type processing.
8. We won't be working with the first sequence anymore, so click on the first sequence window's close gadget (in the upper left corner) to close it.

9. Click on the remaining sequence window. Select the **Save As...** menu item under the **Project** menu, and save the sequence as "**PingPongImages.seq**" in the same directory as "**SampleImages.seq**".

Now, we're ready to specify how to process our images.

4.6.3 Processing the Images

Let's now tell FRED which scripts to use for processing.

1. Select all of the cells with the **Select All** menu item (under the **Edit** menu).
2. Select the **Process...** menu item (under the **Scripts** menu).
3. Add the following scripts in the order listed below:
FREDSCRIPTS:FREDOperators/ScaleToSize.fred
FREDSCRIPTS:FREDOperators/SimpleRotate.fred
FREDSCRIPTS:FREDOperators/DisplacePixel.fred
FREDSCRIPTS:FREDRenderers/RenderAsDefined.fred
FREDSCRIPTS:FREDSavers/SaveAsANIM.fred

These scripts will scale each image to the same size, incrementally rotate them, incrementally then decrementally displace its pixels (to give that explosion/implosion look), render each frame, and save it as part of an ANIM file.

4. Select the **Process** button to begin the processing.
5. Answer the questions posed to create frames that are 320 x 200, with a full 360 degrees of rotation, rendered in low resolution, non-interlace, NTSC, 32 colors, no dithering, and saved as a BYTE type of animation with Faster compression, wrapped file, and palette locked to the first frame. You can specify to save the animation file to **RAM:** as "**Explode.anim**". We will want an explode/implode effect to be done to our images.
6. Once the animation has been created, view it with any standard ANIM viewer.

4.6.4 Conclusion

In this tutorial you have learned how to:

- copy and paste cells within and between sequences
- create a ping-pong effect
- scale all images to the same size
- build complex effects from simple effects

4.7 Lesson 7. FRED: Creating an ANIM From a Sequence of Images

Level: Beginner

Many users of ADPro or MorphPlus will want to create animation files (in ANIM format) from a sequence of images. This tutorial will show you how you can use FRED to do so.

4.7.1 Setting Up

If FRED is not already running, double-click on its icon. A custom screen should appear after a few seconds of disk activity.

Be sure to run ADPro or MorphPlus as well, if it's not already running in the background.

4.7.2 Specifying Your Image Sequence

FRED works on sequences of images. An animation is also a sequence of images. To create an animation (in IFF-ANIM format), you need to create what is called a "sequence" in FRED, containing references to your images.

Each sequence is visually represented in its own window, where each image reference is a cell (a raised box) within this window. Internally, these FRED sequences are plain ASCII text files with the filenames of the images, along with other information, such as the image's dimensions and number of frames represented by this image (see the chapter in the User Manual on FRED for more information on this number).

To create our sequence of images to process into an ANIM, do the following:

1. Select **Open...** from the **Project** menu to open a new sequence window.
2. Select **Image(s)...** from the **Insert** submenu under the **Edit** menu to display a file requester. It is from this requester where you will select the images to insert into the sequence window.
3. Select the drawer where the images are located.
4. Select the image file that you want to insert into the sequence. You can either click on its name and select the **OK** button, or, more simply, double-click on the name—both do the same thing.

This is fine for just one image, but it would be too much work to do this operation for each image you want in the animation. Let's use multiple-selection:

1. Open the Insert Images file requester again, changing to the directory where the images are located.
2. Hold down the **Shift** key and click on the images you want to add to the sequence.
3. Select the **OK** button to add the images to the sequence.

4.7.3 Building the List of Scripts

Before we select the scripts to use, we must tell FRED which cells (images) we want to process. This is done by selecting them. Since we want all the cells to go into our ANIM, we just select all of them. Choose the **Select All** menu item under the **Edit** menu. Notice how the text backgrounds on all the cells has turned green. This color signifies that a cell is “selected”.

The next step is to specify the list of special ARexx scripts for FRED (named FRED Scripts) that will modify this sequence of images into an ANIM file. Although you can create you own scripts to do this processing, FRED already comes with the necessary scripts.

Let's select these scripts.

1. Select **Process...** from the **Scripts** menu to display the Invoke ADPro control panel.
2. Select the **Add...** button to display the file requester.
3. From this file requester, change the **Drawer** to “**FREDSSCRIPTS:**”.

This assignment should've been made at the time you installed the ADPro and FRED ARexx scripts. If not, the program will ask you to insert the “**FREDSSCRIPTS:**” volume. If this happens, do the following:

- (a) Flip back to the Workbench screen by either clicking on the FRED screen's front/back gadget (in the upper right corner of the screen) or holding down the **Left-Amiga** key and pressing the “**M**” key to cycle through the open screens until you get to the Workbench screen.
- (b) Open a new Shell by clicking on the Shell icon. This icon is originally located in the “System” drawer of your Workbench, System, or Work partition.
- (c) Type the following command at the Shell prompt, then press the **Return** key:

```
assign FREDSSCRIPTS: directory_of_FRED_Scripts
```

- (d) Flip back to the FRED screen.
- (e) Click on the file requester to reactive it.

4. A subdirectory named "**FREDRenderers**" should be visible. Click on it to display the scripts within it.
5. Select the "**RenderAsDefined.fred**" script to add it to the script list. Since the ANIM format does not support raw image data, we need to add this script to the script list so that rendered image data can be created.
6. Select the **Add...** button again, this time changing the **Drawer** from "**FREDRenderers**" to "**FREDSavers**". Do this by selecting the **Parent** button, then selecting the "**FREDSavers**" subdirectory.
7. Select the "**SaveAsANIM.fred**" script to add it after the "**RenderAsDefined.fred**" script.

What these two scripts are doing is, for each frame, creating rendered data and appending it to an ANIM file; if it is the first frame, then the ANIM file is created (this is done automatically by the "**SaveAsANIM.fred**" script).

Next, we need to set the **Invoke Scripts:** and **Load Image:** cycle gadgets to the intended settings.

1. Click on the **Invoke Scripts:** gadget until it says, **Once Per Cell**; actually, it can be set to **Once Per Frame** since each cell represents one frame (i.e., length = 1).
This setting means that the each script in the script list will be invoked for each image (cell).
2. Click on the **Load Image:** gadget until it says, **Every Frame**; actually, it can be set to **Every Cell** for the same reason as above.
This setting means that the original image will be loaded into ADPro or MorphPlus (whichever one you're using) for each "pass" through the script list.

4.7.4 Instructing ADPro or MorphPlus to Process Your Images

Now all that we need to do is start the processing, answering the questions posed, and sit back and wait.

1. Select the **Process** button to initiate the processing.
2. Although you might want to watch the progress indicator on FRED's screen, you might want to consider flipping the ADPro or MorphPlus screen to the front while processing is happening. Since the ADPro/MorphPlus screen will, by default, be a lower resolution than FRED's, it takes less time to refresh the screen.

What will happen is FRED will go through your script list and check for the existence of a pre-script to execute. These pre-scripts have a similar name as the scripts themselves, except that they have “.pre” appended to them.

The first script, “**RenderAsDefined.fred**”, requires a few questions be answered and settings selected. ADPro’s or MorphPlus’ screen will appear with questions regarding horizontal and vertical resolution and dithering. What you should select depends upon how you want your images displayed.

The second script, “**SaveAsANIM.fred**”, will ask you the name of the ANIM file to create, as well as the type of ANIM (ANIM-5 or ANIM-8) to produce.

After you answer all of these questions, the FRED screen will reappear, with ADPro or MorphPlus doing the work for you in the background. You will see the progress of the ANIM creation by the indicator on the FRED screen. At any time you can, for interest sake, flip to the ADPro/MorphPlus screen to see what it is doing.

Once it is done, control will return to the FRED screen. You are now ready to view your animation.

4.7.5 Viewing Your Assembled ANIM

The ADPro package comes with a picture/animation viewer called View, but you can use any ANIM-compatible viewer (unless you created an ANIM-8 animation, which will require a compatible player).

4.7.6 Conclusion

In this tutorial you have learned:

- what a FRED sequence represents
- how to multiply-select image files
- where the pre-written FRED scripts are located
- what scripts are required to create an ANIM file
- how to set the Invoke ADPro controls to get the desired results
- why you might not want FRED’s screen up front during processing

4.8 Lesson 8. FRED: Creating a Fade Between Two Images

Level: Intermediate

The best way to learn FRED is to use it. In this tutorial you will create an eight frame animation consisting of a fade between two images. To run this tutorial, you will need:

- ADPro and FRED
- Two images from which the animation will be created
- Free hard disk space amounting to approximately 8 times the size of the larger of the two selected images

4.8.1 Setting Up

Double-click on FRED's icon to start it up. After a short amount of disk activity, the FRED screen should appear.

Be sure FRED's backdrop window is currently selected. This will be apparent when the borders along the edge of the screen change to a green color.

4.8.2 Creating a Sequence

Before we can start composing images with the Compositor, we first need two sequences of images, where each sequence contains just one image. The Compositor requires two sequences.

From within FRED, select the **New...** menu item from underneath **Project**. This opens a new sequence window for editing.

Notice that the **New...** menu item had a reversed, slanted "A" followed by an "N". This letter is a keyboard short cut for selecting this menu item. Using keyboard equivalents allows you to keep your hands on the keyboard, without having to reach for the mouse to perform an operation. Many of the other items have these equivalents. We will use these keyboard short cuts later in the tutorial.

Also notice how the sequence window's border is now colored in green. This is an indication of the currently active (selected) sequence. This is an important point to remember when working with more than one sequence at a time.

4.8.3 Adding an Image to a Sequence

Each sequence contains zero or more images. This tutorial requires two sequences, each with one image.

1. To insert an image into the currently selected (shown with green borders) sequence, select **Image(s)...** in the **Insert** submenu under the **Edit** menu. A file requester will appear from which you can select an image. Please select any color IFF-ILBM image.
2. You can either double-click on the filename to automatically select it, click on it and press the **OK** button, or type the filename in the **Drawer** and **File** input fields and press the **Return** key.

Notice how a black cell is added to your sequence window. This indicates that one image is currently part of this sequence. Actually, the actual image is still on disk, but a reference to it has now been defined.

4.8.4 Making Stamps for Images

NOTE You must have ADPro running before proceeding. Please start that program by either double-clicking on its icon from the Workbench screen.

1. To see a stamp-size representation of your selected image, click once on the black cell (notice how the border around the text below the cell is “pushed in”).
2. Press the keyboard sequence, **Right-Amiga + M**. This is the keyboard equivalent for the **Make Stamp** menu item (under **Animation**).

After some time you should see a small preview of your image within the cell. What has happened is ADPro has loaded in the selected image, scaled it down to stamp size, saved out this “stamp” within the same directory as the original image (but with a file extension of “.stp”), and displayed within the sequence window.

4.8.5 Saving Sequences

Now that we have created our first sequence, we must save it to disk for use later.

1. Press the keyboard sequence **Right-Amiga + A (Project, Save As...)**. A file requester will appear showing the current contents of the

ADP_FRED:Sequences directory. Although you can change this directory to whatever you want, we will save our sequence file to this directory for simplicity.

2. Enter the name **SEQUENCE1.seq** in the **File** input field and press the **Return** key. After a short amount of disk activity, the name of the sequence (shown on the title bar) changes from **Untitled0.seq** to **SEQUENCE1.seq**.

Now that this sequence has a name, if you want to save the latest changes, you can simply select the **Save** menu item instead.

4.8.6 Creating a Second Sequence

We now need to prepare the image we're going to fade to.

1. Using the knowledge you attained from the previous steps, create a second sequence and insert a color image, different than the one for the first sequence. You may create a stamp for this sequence's image if you want.
2. Save this sequence under the name of "**SEQUENCE2.seq**".

4.8.7 Launching the Compositor

Now that we have our two sequences—background and foreground images—we are ready to create a fade between the two.

1. Click the mouse pointer anywhere within FRED screen, but outside of any sequence windows currently visible on it, to activate FRED's backdrop window. You will know that you have done this when both sequence windows become deselected and the backdrop window's borders turn green.
2. Select the **Compositor** menu item from the **AnimOps** menu. This displays the Compositor AnimOp in its "No Project" mode—a thin window with the words "ADPro Sequence Compositor (No Project)" on its title bar. Notice how this AnimOp opens its window on top of FRED's screen. Almost all AnimOps work this way.
3. Create a new Compositor project by selecting **New** from the **Project** menu. The thin window will expand into the Compositor's Define Components control panel, which is where we define the components of the fade.

4.8.8 Specifying Background and Foreground Sequences

The first time you open the Define Components control panel it is ready to define the background sequence of images (sequence 1, as displayed in the current sequence indicator at the bottom of the panel).

1. Press the **Sequence** button.
2. Select the sequence called **SEQUENCE1.seq**.
3. Our fade between the two images will be done over eight frames, so we need to have this sequence's image last for eight frames. This is done by setting the **Duration:** value to 8 (if it isn't that already) and the **Underflow:** cycle gadget to **Repeat**. This means that the last image (actually, the only image) in the sequence will get used over and over again, which is what we want.
4. The other values in this control panel should be set to the following (which should be their default values):

Alpha...	not defined
Pre Load Hook...	not defined
Post Load Hook...	not defined
Loader...	UNIVERSAL
Loader Options:	not defined
Start X:, Y:	0, 0
End x:, y:	0, 0
In Frame:	1
Skip:	0
Offset:	0
Trans Color R:, G:, B:	-1, 0, 0
Mix From:, To:	0, 0

This is the background sequence. Now we need to define the foreground sequence.

1. Select **Add Sequence** from the **Edit** menu. Notice that the current sequence indicator now says "Sequence 2 of 2".
2. For this sequence, enter the following values:

Sequence...	SEQUENCE2.seq
Alpha...	not defined
Pre Load Hook...	not defined
Post Load Hook...	not defined
Loader...	UNIVERSAL
Loader Options:	not defined
Start X:, Y:	0, 0
End x:, y:	0, 0
In Frame:	1
Duration:	8
Underflow:	Repeat
Skip:	0
Offset:	0
Trans Color R:, G:, B:	-1, 0, 0
Mix From:, To:	0, 100

Note that the **Mix** level goes from 0 to 100. This means that the images in this sequence will start off as not being visible and ending with the entire image being totally visible. This will give us the effect of fading from background sequence to foreground sequence.

You can switch between the two sequences by using the **Next** and **Previous** menu items under the **Edit** menu.

4.8.9 Specifying the Resulting Sequence

Now that we have defined the components, we must define the result sequence and frames that will be generated.

1. Select the **Define Results** subitem from the **Project, Mode** menu item. This is the Results control panel.
2. Press the **Output Sequence...** button.
3. Enter the name "**RESULT.seq**".
4. Press the **OK** gadget from the file requester. This will create a sequence file that can be loaded back into FRED or any other program that can utilize FRED sequence files.

Each of the eight frames that will be generated will have a particular base or root name associated with it. This name is used to assemble the file-name by appending the base name with the current frame number (i.e., **Work:Images/TestPic.00000**, **Work:Images/TestPic.00001**, and so on are produced by entering **Work:Images/TestPic** as the base name). Note that these images start with a zero frame number.

5. Enter a base name for your images. Please do not place embedded spaces (i.e., "**Work:Images/Test Pic**") within your filenames. These

spaces may confuse the program and cause it to be handled improperly with some programs.

6. We want to generate eight frames that show the fade from one image to another, so enter the number 8 into the **Total Frames:** field.
7. Enter the following information into the other fields:

Pre Save Hook:	not defined
Post Save Hook:	not defined
Pre Script Hook:	not defined
Post Script Hook:	not defined
Saver:	IFF
Saver Options:	RAW
Skip Frames:	0
Process Frames:	8

4.8.10 Processing the Images

To start the processing of the images, select the **Accept** menu item from the **Project** menu. A meter window will appear atop the FRED screen showing the progress of the compositing operations.

Once the meter indicator moves all the way to the right side of the window and the window disappears, you can examine the processed images.

You can close the Compositor by selecting the **Project, Quit Compositor** menu item. A requester will pop up telling you that changes have been made to this project, but you can ignore this warning by pressing the **Close** button. In this tutorial, we are more interested in the results themselves. If you would like to save this project for future examination, please press the **Save And Close** button instead and type in the name you want to give this Compositor project.

4.8.11 Displaying the Fade

To see the fade, you must first open the sequence.

1. Select **Project, Open...**
2. Double-click on the sequence file called **RESULT.seq**. Another sequence window will appear, but with eight black cells in it.

Generate stamps for all of these cells. First, you will need to select the images that you want to modify.

1. Select the **Select All** menu item from the **Edit** menu. Notice how all of the text boxes below each cell now has a background color of green. Black text on a green background signifies that a particular cell is part of the selected group of cells.
2. Create stamps for these cells using the steps you learned before.
3. To display the fade as a stamp-size animation, select **Ping Pong...** (a subitem of **Animation, Playback**). A small window will appear atop the FRED screen with the fade in motion. If you find that the images are flickering, drag the window down to the bottom of the screen. This usually reduces the flickering.
4. You can change the frame rate at which each frame is displayed by making another choice from the **Speed** submenu, also found under the **Animation** menu.
5. To stop the animation, click on the preview window's close gadget.

You should now have a series of images (located in the directory defined by the **Image Root Name...**) that display this fade.

This step concludes this lesson.

4.8.12 Conclusion

In this tutorial you have learned how to:

- create, load, and save sequences
- use menu item short cuts to speed up your work
- add images to your sequences
- use an animation operator (AnimOp) to modify two existing sequences
- animate stamp-size representations of your images

4.9 Lesson 9. Sentry: Automated JPEG Archiving

Level: Intermediate

The Sentry can be used to automate the process of archiving images. Video Toaster users should find the following lesson very useful for archiving their FRAMESTORE or RGB (IFF24) files in JPEG format.

This lesson requires that you have the following items available:

- The **SaveAsJPEG.fred** FRED Script, which is automatically installed if you select to install the FRED ARexx scripts during the ADPro or MorphPlus installations.

These items will be used during the course of this lesson. If you do not have all of these items, do not proceed with this lesson.

4.9.1 Setting Up

In this lesson, we will be monitoring a directory where Framestores are located or to where they will be saved. We will also need enough free disk space to save at least one JPEG'd file.

1. Flip to the Workbench screen by either pressing the current screen's front/back gadget (the upper-rightmost gadget on the screen) or, if the screen doesn't have this gadget, hold down the **Left-Amiga** key and keep pressing the "M" key until the Workbench screen appear.
2. Double-click on the Sentry icon. The main Sentry control panel will appear, ready for a new project to be defined.

4.9.2 Specifying the Directory to Monitor

We first need to specify the directory to monitor.

1. Double-click on the **Directory...** button.
2. Select the directory where your FRAMESTORE or RGB files are currently stored or where they will be saved (remember that Sentry can process files as they are generated). Select the file requester's **OK** button to confirm your selection.

Notice how the selected directory is displayed within the input field to the right of the **Directory...** button.

3. Click on the **Scan Mode:** cycle gadget until it displays the phrase, **Continuous** (if it doesn't display it already). This tells Sentry to continue processing files until you tell it to stop.

4.9.3 Filtering the Files to Process

If the files in the selected directory have (or will have) a certain type of filename (such as: *number.FS.text* for most Framstore files), then tell Sentry that you only want those named files to be considered.

1. Enter a filename pattern in the **Include Pattern:** input field. For the example pattern above you would enter:

#?.FS.#?

which means to consider any file that has “.FS.” somewhere in the middle of its filename.

Note that this is not as strict a pattern as can be, but should be simple and easy enough to remember for most users. If you want to process all the files in the selected directory, then use: **#?** for the pattern.

2. The **Exclude Pattern:** specification can be set to another filename pattern, although leaving it blank for this lesson should be fine. This value operates similarly to **Include Pattern:** in that it filters filenames, except in this case it removes files from being considered for processing.
3. If the selected directory already has files within it that need processing, then make sure the **Consider Existing Files?** checkbox is checked.
4. If you want the processed files (these are the original files, and not the JPEG files that will be generated) to be deleted after being used, check the **Delete Processed Files?** checkbox.

You might want to do this if you are low on disk space or if you will be low if both the original and created versions of the files are on the disk at the same time. If this checkbox is checked you can control whether Sentry will ask you to confirm the removal of the processed files by checking the **Confirm Before Deleting?** checkbox.

4.9.4 Specifying How to Process Files

To actually tell Sentry what to do with the files, we need to define a list of “steps” or commands to perform on the files. This is done through the list at the bottom of the panel.

1. Select the **Add...** button to enter a new command specification.

2. Make sure the **Type:** cycle gadget is set to **Rexx**. We will be using one of the supplied FREDScripts—namely, the **SaveAsJPEG.fred** script—for use in our project.
3. Select the **Command...** button.
4. Locate the **FREDSCRIPTS:FREDSavers/SaveAsJPEG.fred** script using the file requester controls. Select the file requester's **OK** button to accept that script.
5. Make sure the **Arguments:** input field is blank.

4.9.5 Processing JPEG Files

Now we are about ready to process the JPEG files.

1. Before we proceed, save the project (use the **Save As...** menu item under the **Project** menu) under the name: **ArchiveAsJPEG.spj** (the “.spj” is short for “Sentry Project”).
2. To perform the operation, select **Accept...** under the **Project** menu.

The Status panel will appear, along with another smaller panel (the Number Of Cells panel) above it. Sentry noticed that you specified a FREDScript. These types of scripts need to know how many files (cells) will possibly be encountered.

1. To process as many files as possible, enter 0. This value means that we have no idea how many we have and to just process as many files as possible.
2. The **SaveAsJPEG.fred** script will then ask you some questions relating to how the files should be JPEG'd as well as how to modify the files' names. For general use, you can use the default value of 32 for quality level. When it asks how the filename should be modified, select **Append ???** and accept the default “.jpeg” filename extension. Each JPEG'd file will then have .jpeg appended to it.

NOTE You should choose to move these JPEG'd files to a different directory. **THIS IS VERY IMPORTANT!** Keeping the modified files in the same directory will cause Sentry to JPEG them again, which is not what you want at all.

You can stop processing by either selecting the **Control Panel...** or **Quit** buttons.

After processing has completed, your images should then be archived in JPEG format, with little work on your part.

4.9.6 Conclusion

In this tutorial, you have learned:

- how to use Sentry to monitor a directory for new files.
- how to filter filenames which should be processed.
- how to utilize the numerous FREDScripts available within the ADPro package.
- what is required to use FREDScripts with Sentry.

We hope that this short lesson gives you a flavor of what can be accomplished with Sentry.

Chapter 5

Using ADPro

This chapter will describe all aspects of using ADPro from its graphical user interfaces, or GUIs. Detailed descriptions are included for loading, saving, processing, and printing images, as well as executing ARexx scripts.

5.1 Overview

This section gives a quick overview of starting, displaying information about, and exiting ADPro.

NOTE All references to Art Department Professional (ADPro) apply to the MorphPlus program, unless otherwise indicated. If you currently own and use MorphPlus, note that both ADPro and MorphPlus cannot be running at the same time—either one but not both. If you will be using MorphPlus, then disregard this subsection on starting ADPro.

5.1.1 Starting the Program

You can start the ADPro program from either the Workbench or Shell.

On startup, if **NOLOADSETTINGS** was not specified, ADPro will look for its settings file in the following directories (in the listed order). It will use the first one it finds:

1. The settings file specified with the **SETTINGS** icon Tool Type or Shell

argument.

2. The settings file named “ADPro.prefs” in the same directory as the program.
3. The settings file named “ENV:ADPro/ADPro.prefs”.
4. Use the default (built-in) settings.

Workbench Startup



For Workbench users, open the directory where the ADPro program is located. Locate this icon as shown in the left image above. It should have the name **ADPro** below it. Double-click on this icon to start the ADPro program.

Shell Startup

Before starting the program, either be in the directory where ADPro is currently located or make sure you have previously executed the command:

```
Assign ADPRO: location_of_ADPro
```

This will allow ADPro to find all of its support files. A good place to put this command is in your **S:user-startup** file, if it hasn't already been placed there by the installation utility.

For Shell users, change the current directory to the directory where ADPro is located and start the program using the two commands below. Type each command on its own line, pressing the **Return** key after each line.

```
cd ADPro-directory  
ADPro
```

ADPro-directory must be replaced with the actual directory name where the ADPro program and its files were installed. Note that you can prepend the **run** command before **ADPro** above so that you can continue to use the Shell while ADPro is running.

Various settings can be specified on the same command line as the program. See Section 13.1 for more information on these settings.

If ADPro is successful in initializing itself, then you will see the ADPro window. If no window is displayed, you might have run out of available memory for the program to use or the program could not find a file it needed. In either case, consult Appendix A (Troubleshooting) for more assistance.

5.1.2 Displaying Information About ADPro

Selecting the **Release Notes...** menu item (under the **Project** menu) will display the text file containing information about last minute changes to the package.

The **ReleaseNotes** utility (located in **ADPRO:**) is launched to view the text file “**ADPRO:ADPro.notes**” using the text viewer specified in the **ReleaseNotes** utility’s icon. The text will be displayed on the same screen as ADPro, if the text viewer can be directed to do so. If no text file is present, then a file requester will appear for the selection of one. The **ReleaseNotes** utility is described more in Appendix E.4.

Selecting the **About...** menu item (under the **Project** menu) will display a window with a scrolling list of information, including:

- the revision and serial number of this copy of ADPro
- the size (in bytes) of ADPro’s primary image buffer
- an indicator of whether the “temp” buffer (see Section 3.2.2) contains image data
- the name of the ARexx port
- text asking for your cooperating in keeping ADPro professionally supported

5.1.3 Exiting the Program

To exit the program, select the **Quit ADPro** menu item (under the **Project** menu) or select the main window’s close gadget.

On exit, if **NOSAVESETTINGS** was not specified, ADPro will store the current program settings under the first condition that is met (it will try the next numbered condition if the current condition fails to save the settings):

1. The settings file specified at startup.

2. The settings file named “ADPro.prefs” in the same directory as the program.
3. The settings files named “ENV:ADPro/ADPro.prefs” and “ENVARC:ADPro/ADPro.prefs”.

This allows you to retain many of the program settings from session to session.

5.2 Using the Controls

This section gives an overview of the controls available from the ADPro interface. Both the List and Button Interfaces will be described, as well as how the floating module lists can be used with each interface.

5.2.1 The List Interface

ADPro can be run as a sizeable window comprised of scrolling lists for the loaders, savers, operators, and user commands. This list is very useful when you want visible access to all the modules.

To switch to List Interface mode, unselect the **Button Interface** menu item under the **Settings** menu.

As shown in Figure 5.1, the List Interface contains vertically scrollable lists of modules and information. This interface is useful for people who want easy access to all the available modules. It is suggested that this interface only be used on interlace screens because of the screen space it uses; see Section 5.11.1 for information on setting ADPro’s screen mode.

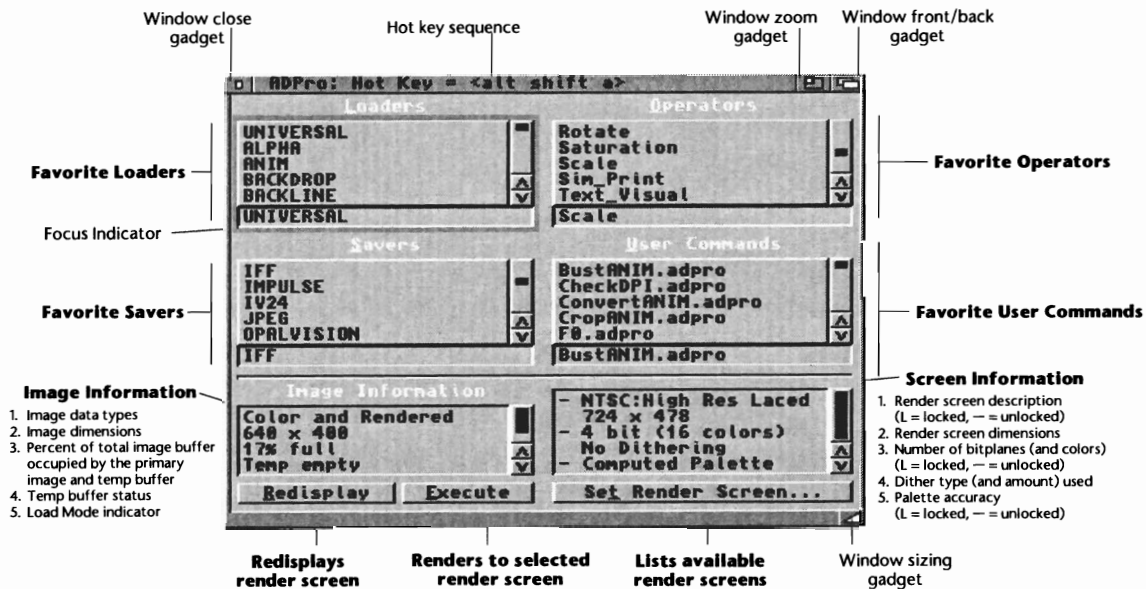
The **Loaders** list contains all the accessible loader modules; similar for the **Savers** (saver modules), **Operators** (operator modules), and **User Commands** (ARexx scripts) lists.

The **Image Information** list displays the currently loaded image’s data types, the width and height in pixels, and the percentage of the image buffer occupied by this image. The **Redisplay** and **Execute** buttons below this list control the display of the rendered image data.

The Screen Information list displays the currently selected render screen and the dithering to use. The **Set Render Screen...** button below this list allows you to specify the render screen to use.

The thick outline around one of these lists is the input focus indicator. This indicator is needed so you know which list is currently “active” to accept scrolling and activation from the keyboard. For instance, to scroll through the

Figure 5.1: The main ADPro controls in the List Interface layout.



available list of loaders, move the focus indicator to the **Loaders** list (press the **Tab** key to change focus to this list) and press the cursor up and down keys. Notice now the list scrolls. Now, if you pressed the **Tab** key again (this moves it to the **Operators** list), then pressed the cursor keys, the operators list will scroll. You can also use the **Return** key to launch the current module.

NOTE The focus indicator is only available in the List Interface.

5.2.2 The Button Interface

ADPro can also be run as a non-sizeable window comprised of buttons for your favorite modules.

To switch to Button Interface mode, select the **Button Interface** menu item under the **Settings** menu.

As shown in Figure 5.2, the Button Interface contains groups of buttons for the loader, saver, operator, and user command modules, as well as vertically scrollable lists of information (the same as those used in the List Interface). This interface is useful for people who want easy access to their most favorite modules, and who want to know their keyboard equivalents without searching for them in the menus. This interface can be used on either non-interlace or interlace displays; see Section 5.11.1 for information on setting ADPro's screen mode.

The **Loaders** buttons show your two favorite loader modules as well as the control for selecting the orientation and compositing mode; for the **Savers**, **Operators**, and **User Commands** buttons, three favorite modules are shown. These favorite modules are the same as what are listed in the menus.

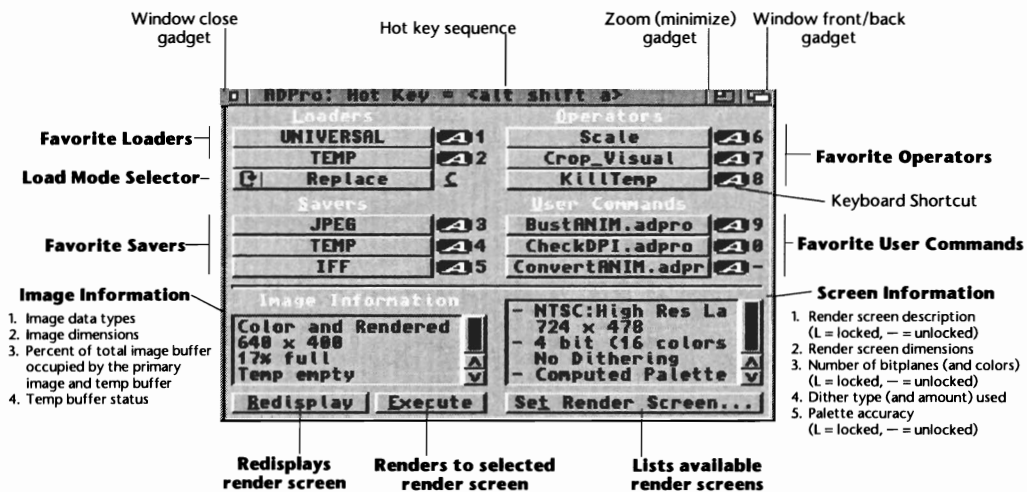
The **Image Information** list displays everything as in the List Interface except the Load Mode indicator, which is as its own cycle gadget in the window. The Screen Information list displays the same information.

Customizing the Buttons

To set one of the "favorite" buttons or menu items to a specific module, do the following:

1. Open the module list (see Section 5.2.3 for how to do this) that corresponds to the type of module button to set.
2. Click once on the name of the module.
3. Hold down the **Shift** key and click with the left mouse button on the button to assign the selected module.

Figure 5.2: The main ADPro controls in the Button Interface layout.



Alternatively, you can select the **Module #*n*** menu item (where *n* is **1**, **2**, or **3**) from the **Set Modules** submenu under the **Settings** menu. This method has the advantage over clicking on the buttons in that you can assign a module even if the corresponding button is disabled (which will happen to the Saver and Operator buttons when no image data exists in the primary image buffer).

Note that you cannot assign a module of one type to a button of another type. If you try, then the operation will be ignored and the module currently assigned to that button will be invoked.

5.2.3 Floating Module Lists

In either the List or Button Interface, you can open up windows with vertically scrollable module lists in them. These lists are similar to those in the List Interface, except that each list is in its own window that can be moved and resized independent of the other list windows. In addition, you can invoke modules from either these module list windows or from the main window interface, be it Button or List. As such, these module lists are termed “floating” or asynchronous because they appear in windows which can be opened, closed, repositioned, and resized independent of the main window.

These floating module lists are sometimes called “floating tool palettes” on other platforms, but we will use the term “floating module lists” to avoid confusion with ADPro’s palette operations.

To open a module list, either select the **Open *module_type* List...** menu item under the particular menu or press the module type’s keyboard equivalent which is the letter underlined in the interface’s module areas (i.e., “**L**” for **Loaders**, etc.) Once open, you can move and resize it just like any other window. Notice that when one of these lists is open, its menu item changes from **Open ...** to **Close ...** to tell you that its window is currently open; this is handy if you are running ADPro on a large or crowded screen.

You can scroll the list using the slider knob, arrow buttons, or cursor keys. These function the same as all other lists within ADPro.

NOTE Note that if you select an item, scroll the list with the scroll bar so that the selected item is out of view, then use the cursor keys on the keyboard to select the previous or next item, the list will readjust itself so that the new selected item is within view. These floating lists always try to keep the selected item within view.

To invoke a particular module, either double-click on its name in the list or select it and press the **Return** key (which launches the currently selected module).

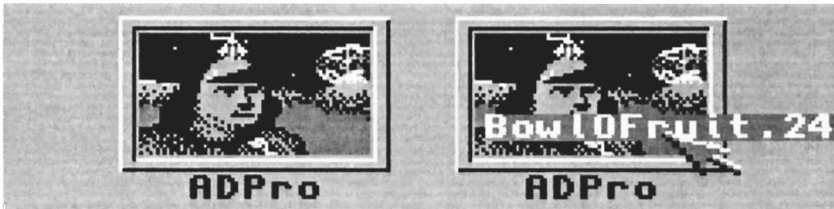
To close a module list, select the **Close** *module_type* menu item, click on the list window's close gadget, or, with the list window active, press the **Esc** key.

You might find these floating module lists useful when you need to use several modules (e.g., operators) in a given session but don't want the other module lists visible. In this example, the Button Interface wouldn't give you the easy selection of all operator modules and the List Interface wouldn't give you the simplified interface. You can use either interface for the main window, but will probably want to minimize the window (use the **Minimized** menu item under the **Settings** menu) and move it to a corner of the screen, then open the operator list along either side of the screen from top to bottom.

5.2.4 AppWindow, AppIcon, and AppMenu Support

When ADPro is running on the Workbench screen, its window becomes an AppWindow. This allows you to drag-and-drop file icons onto it and have them processed via an ARExx script. The script that it tries to execute is defined by the **APPSCRIPT** tool type or **AS** Shell argument. See Section 13.1 for more information on using them.

When ADPro is running on its own custom screen or on another public screen, an AppIcon appears on the Workbench screen. This works similar to the AppWindow, in that you can drag-and-drop file icons onto this AppIcon. You can perform any operation on these dropped files by starting ADPro with the **APPSCRIPT** tool type or Shell argument, as described above.



If you double-click on this AppIcon, the ADPro screen and window will move to the front of all other screens and windows.

In addition to the AppIcon, an AppMenu named “**ADPro**” will appear in the **Tools** menu on the Workbench screen. Selecting this menu item will bring ADPro's screen and window to the front. Also, you can manipulate files just as in the AppWindow and AppIcon cases by a slightly different method. As there is no “container” to drop a file on, what you would do is select the file or files to “drop”, then select the AppMenu item. The **APPSCRIPT**-defined script will be invoked on them.

5.3 Loading an Image

Before you can work on an image you must first load it into the ADPro program. This section will give a general description of how to do this. If you need more information about a specific loader, please consult Chapter 6.

5.3.1 Loader Module

The process of importing image data to ADPro's image buffer is handled by programs or modules called "loaders". Loaders not only import data from files on disk, but they can also *create* images. Basically, the connection between an external source of image data and ADPro's image buffer would be handled by a loader.

There are several ways of invoking a loader, depending upon the type of user interface you're using.

With the List Interface, scroll the **Loaders** list until the loader module's name appears within view, then double-click on the name. Each loader will act differently, so consult Chapter 6 for the specific controls. Note that the UNIVERSAL loader is always at the top of the list for convenience.

With the Button Interface, invoking a previously-set loader button is as simple as clicking once on the button corresponding to the loader or pressing its keyboard equivalent.

With either interface, you can open the floating Loaders list and double-click on the module name. Any change (scrolling or selecting) you make to the List Interface **Loaders** list automatically appears in the floating **Loaders** list, and vice versa.

The **Open...** menu item under the **Project** menu is a special command which will use the UNIVERSAL loader to specify a file to load. Selecting this command acts the same as if you invoked the UNIVERSAL module, except that the currently selected loader in the **Loaders** lists are not affected. This command is available as a menu command because loading files of a particular format from disk is a common operation. It is named **Open...** to convey the importing of a "project file" (in this program, an "image") from disk.

When you use the UNIVERSAL loader, you might not know what loader module ADPro is trying to use to interpret the image data. As an aid, the window's title bar will change to, "Invoking *module_name* Loader...", when it is executing a loader module. In fact, a similar message will appear for saver and operator modules, although they are not as useful as when you are invoking the UNIVERSAL loader.

5.3.2 Load Orientation

ADPro can perform a 90 degree counter-clockwise rotation of the image data *during a load operation*. This is accomplished with the **Orientation** submenu of the **Loader** menu.

Note that the **Orientation** setting affects data *only during a load operation*.

Using a combination of the **Horizontal_Flip** and **Vertical_Flip** operators and the **Orientation** setting, ADPro can produce 90 degree rotations through 0, 90, 180 and 270 degrees.

These can be performed as follows:

0 Degrees	Load the image in the Portrait orientation.
180 Degrees	Load the image in the Portrait orientation and then perform both a horizontal and vertical flip. The resulting data will be rotated 180 degrees with respect to the data loaded from disk.
270 Degrees	Load the image in the Landscape orientation. The resulting image will be rotated 270 degrees with respect to the data residing on disk.
90 Degrees	Load the image in the Landscape orientation and then perform both a horizontal and vertical flip. The resulting data will be rotated 90 degrees with respect to the data loaded from disk.

NOTE Although you can use the supplied **Rotate** operator to rotate to any degree (much more flexible than the previously mentioned orientation tips), these tips are included for when you want to reorient your image when it is being loaded.

5.3.3 Image Compositing

ADPro supports a very powerful image compositing feature which can be enabled by either setting the Compositing submenu to **Compose** or, if using the Button Interface, selecting the **Compose** checkbox.

The default setting is to disable image compositing. In this case, the submenu selected will be **Replace** and the **Compose** checkbox will be unchecked.

When in **Compose** mode and a load operation is performed, you will be presented with the Image Compositing control screen (shown in Figure 5.3). Unless otherwise stated, all loaders support image compositing.

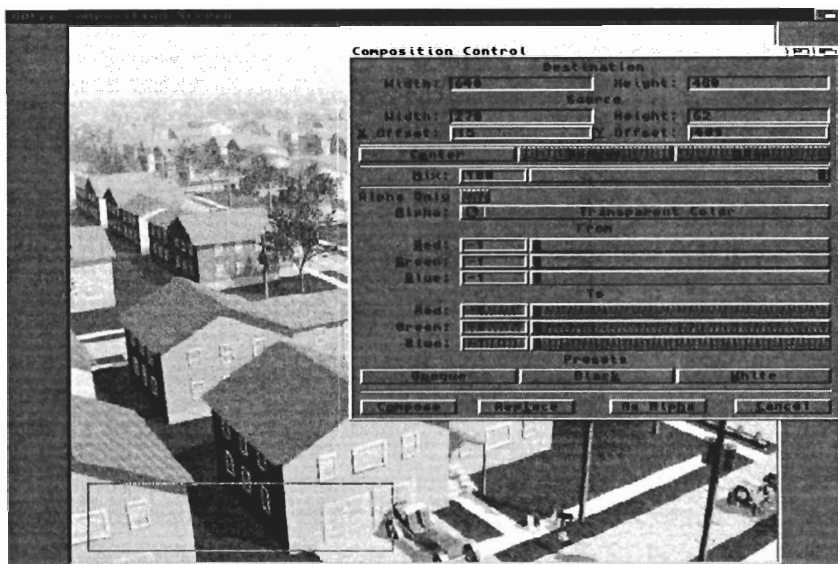


Figure 5.3: The Image Compositing screen and control panel.

Contents Of File	Type Of Raw Data In Memory	
	Gray	Color
Gray	Yes	Yes
Color	No	Yes

Table 5.1: Types of image data that can be merged together.

When performing a non-compositing load operation, the loader discards the previous raw and rendered data before loading new data. When compositing, the loader merges the previously defined raw data with the new image being loaded. The exact balance of the merging operation is under your control.

A file containing a grayscale image can be merged (composited) into grayscale or color raw image data. A color image may *not* be directly merged into grayscale raw image data. Table 5.1 summarizes this restriction.

To merge a color image into a grayscale image, first convert the logical structure of the grayscale image into that of a color image using the `Gray_To_Color` operator.

Keyboard Combination	Movement
Cursor key	-/+ one pixel
Alt + Cursor Up/Down	-/+ one-half of source height
Alt + Cursor Left/Right	-/+ one-half of source width
Shift + Cursor Up/Down	-/+ source height
Shift + Cursor Left/Right	-/+ source width
Ctrl + Cursor Up/Down	Align top/bottom edges
Ctrl + Cursor Left/Right	Align left/right edges

Table 5.2: Key sequences to position the source (foreground) image box.

Positioning Controls

In the Compositing control panel shown in Figure 5.3, the **Destination** sizes correspond to the width and height of the raw image currently in ADPro (i.e., the background image), into which the file to be loaded (i.e., the foreground image) will be merged.

The **Source** width and height denotes the full size of the image to be loaded. The offsets (x and y, labelled as **Offset X:** and **Offset Y:**) give the position of the top left corner of the source image to be loaded relative to the top left corner of the destination image. An offset of 0, 0 means that the top left corner of both the source and the destination will coincide.

Typing offsets into the supplied input fields can be used to set the offset of the top left-hand corner of the image to be merged relative to the top left-hand corner of the original image. Note that negative offsets are perfectly acceptable.

You can use the cursor and qualifier keys to change or “nudge” the offsets, as described in Table 5.2.

Also, you can use the mouse to drag the rectangle representing the source image around on the screen. As you move it, notice how the offset values change instantaneously. You do not need to drag the rectangle itself—dragging outside of the rectangle will do the same thing.

Selecting the button marked **Center** causes the image now being loaded to be centered within the backdrop (previous) image. If the image now being loaded is larger than the backdrop image, its center will coincide with the center of the backdrop image.

If the control panel is getting in your way of positioning the box, you can shrink the control panel to the height of its title bar by clicking on the “zoom” gadget (the gadget to the left of the window’s front/back gadget). It can then

Mix Level	Weight Of Pixels From Loaded Image	Weight Of Pixels From Previous Image
100	100	0
50	50	50
25	25	75

Table 5.3: How mix level affects the new composite image.

been expanded to its normal size by clicking on it again. Alternatively, you can press the "I" key to toggle the size of this window.

Mix and Transparency Controls

The mix level, which can range from 0 to 100, can be set with the **Mix:** input field and slider. It represents the weighting the pixels in the image to be loaded should have when being averaged with pixels from the previous (currently in ADPro) image. Table 5.3 shows how various mix levels affect the weighted average between these two images.

So, for example, a mix level of 100 means that the new image data is given a weighting of 100 percent. Thus, the new pixel completely replaces the old pixel where the new image and the old image overlap. A mix level of 50 means that you will get a straight arithmetic average between new and old pixels.

A specific color in the incoming image can be made completely transparent by setting the **Alpha:** cycle gadget to **Transparent Color** and choosing the transparent color using the **From Red:**, **Green:**, and **Blue:** sliders and input fields. This will allow 100% of the old image data to be preserved for each pixel in the incoming data which matches the color specified.

A range of colors can be made transparent in a similar way. Set **Alpha:** to **Transparent Color Range** and specify the range of color with the **From** and **To** sets of **Red:**, **Green:**, and **Blue:** controls. The **From** values are not required to be less than the **To** values.

When compositing a grayscale image, the value in the **Red:** input field is used for the transparency operation. A value of -1 in any of the input fields disables the transparency check.

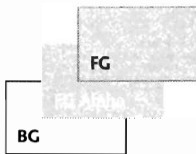
As a convenience, there are three preset buttons which will set the **Red:**, **Green:**, and **Blue:** fields to common values. Use Table 5.4 as a guide to what each preset will do.

Preset	Red:	Green:	Blue:
Opaque	-1	-1	-1
Black	0	0	0
White	255	255	255

Table 5.4: RGB values for the transparent color preset buttons.

Alpha Channel Controls

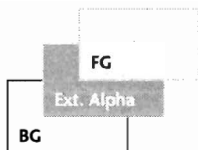
There are various ways of compositing one image over another in ADPro. This section is intended to give you a pictorial and step-by-step guide to all possible options for compositing.



Compositing an image using its embedded alpha channel

1. Load a background image, if one isn't available.
2. Switch to **Compose** mode.
3. Choose to load an image containing an embedded alpha channel.
4. Uncheck the **Alpha Only** checkbox.
5. Set **Alpha:** to **Use Incoming Alpha**.
6. Select the **Compose** button.

ARexx Equivalent: USE_ALPHA



Compositing an image using an external alpha channel

1. Load a background image, if one isn't available.
2. Switch to **Compose** mode.
3. Load the image to be used as the alpha channel.
4. Specify the X and Y offsets of the alpha/foreground images in the **X Offset** and **Y Offset** input fields.
5. Select the **As Alpha** button.
6. Choose to load the image to be composited.
7. Uncheck the **Alpha Only** checkbox.
8. Set **Alpha:** to **Alpha Memory**.
9. Specify the same X and Y offsets as you did above.
10. Select the **Compose** button.

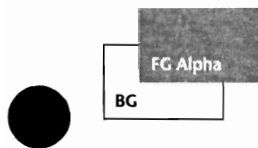
ARexx Equivalent: ALPHA_MEM



Compositing an image (using a transparent color or range)

1. Load a background image, if one isn't available.
2. Switch to **Compose** mode.
3. Choose to load an image to be composited.
4. Uncheck the **Alpha Only** checkbox.
5. Set **Alpha:** to **Transparent Color** or **Transparent Color Range**, and specify the colors in the input fields.
6. Select the **Compose** button.

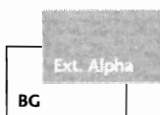
ARexx Equivalent: COMPTRANS or COMPTRANSTO



Compositing using an image's embedded alpha channel

1. Load a background image, is one isn't available.
2. Switch to **Compose** mode.
3. Choose to load an image containing an embedded alpha channel.
4. Check the **Alpha Only** checkbox.
5. Set **Alpha:** to **Transparent Color**.
6. Select the **Compose** button.

ARexx Equivalent: USEALPHA ALPHA_ONLY



Compositing using an external alpha channel

1. Load a background image, if one isn't available.
2. Switch to **Compose** mode.
3. Load the image to be used as the alpha channel.
4. Select the **As Alpha** button.
5. Choose to load the image to be composited.
6. Check the **Alpha Only** checkbox.
7. Set **Alpha:** to **Alpha Memory**.
8. Select the **Compose** button.

ARexx Equivalent: ALPHAMEM ALPHA_ONLY



Replacing with an image's embedded alpha channel

1. Load any image, is one isn't available.
2. Choose to load an image containing an embedded alpha channel.
3. Check the **Alpha Only** checkbox.
Select the **Replace** button.

ARexx Equivalent: ALPHA_ONLY



Replacing with an image

1. Switch to **Replace** mode.
2. Load any image.

ARexx Equivalent: No compositing options specified.

Compositing Commands

Selecting the button marked **Render** causes the background image (the image already in memory) to be drawn in grayscale. This will assist you in determining the composition offsets. In order to display the foreground image (the one you are about to merge) in the same way, ADPro would have to read that image twice (once to create the mock-up for positioning, and once to actually load the image data). Loading an image from an external source (disk file, scanner, tape driver, etc.) twice was thought to be too expensive (in terms of your time), therefore, this capability is not provided. While the background mock-up is rendering, you can stop it using the button marked **Stop**.

The button marked **Compose** will merge the specified image into the currently defined raw image. The button marked **Cancel** will abort the compositing as well as the load operation.

The button marked **Replace** will continue with the load operation but not perform compositing. This will cause the raw image to be completely replaced by the newly loaded data as if you had disabled image compositing.

With creative use of the image compositing capability, a myriad of special effects can be performed. For example, you can use a combination of a standard Amiga paint package and the image compositing function in ADPro to perform true color (24 bit-plane) image masking. Specifically, you can do things such as remove just one person from a scan of a group photo and place that person into the middle of a 24 bit-plane ray tracing.

5.3.4 Locked Screen Mode and Depth

As discussed previously, if ADPro senses rendered image data within a file, it will try to set the render screen to the same mode and depth. In most cases, this is what you want.

But, you might want to set the mode and depth and display the next loaded image in that mode. To avoid any change to the render screen settings, the **Locked Screen Mode** and **Locked Depth** menu items under the **Settings** menu need to be selected.

See Section 5.6 for more information about displaying an image.

5.4 Saving an Image

When you have an image that you like to keep, you should save it to disk. Images can also be exported to devices, such as film recorders and video printers

(with appropriate saver modules). Basically, the connection between ADPro's image buffer and an external destination would be handled by a saver.

You have the choice of saving your image out in one of many image formats. This section will give a general description of how to do this. If you need more information about a specific saver, please consult Chapter 7.

5.4.1 Saver Module

The process of exporting image data from ADPro's image buffer is handled by programs or modules called "savers".

There are several ways of invoking a saver, depending upon the type of user interface you're using.

With the List Interface, scroll the **Savers** list until the saver module's name appears within view, then double-click on the name. Each saver will act differently, so consult Chapter 7 for the specific controls.

With the Button Interface, invoking a previously-set saver button is as simple as clicking once on the button corresponding to the saver or pressing its keyboard equivalent.

With either interface, you can open the floating Savers list and double-click on the module name. Any change (scrolling or selecting) you make to the List Interface **Savers** list automatically appears in the floating **Savers** list, and vice versa.

The **Save As...** menu item under the **Project** menu is a special command which will use the IFF saver to save the image data. Selecting this command acts the same as if you invoked the IFF module, except that the currently selected saver in the **Savers** lists are not affected. This command is available as a menu command because saving files to IFF format is a common operation on the Amiga platform. It is named **Save As...** to convey the exporting of the image data to disk.

5.4.2 Save Data Types

Recall from Section 3.2.1 that ADPro can maintain up to two different types of image data internally: rendered and raw.

For the rendered case, there may be two choices: cropped screen and entire image size. "Cropped screen" refers to the current screen mode's visible area. You can also term this "one screen-full". If the image is smaller than the screen it's being displayed on, then the image area is saved and not the screen area.

“Entire image” refers to the area defined by the image’s pixel dimensions (i.e., the whole image, and not just what is visible in one screen-full).

Most savers will give you an opportunity to select which type of image data you wish to save. Each saver knows what type of image data it will accept. If you do not have the appropriate type of data available at the time the saver is executed, it will alert you to this requirement.

5.5 Operating On an Image

An image can be operated in several ways. This section will give a general description of how to do this. If you need more information about a specific operator, please consult Chapter 8.

5.5.1 Operator Module

The process of manipulating the image data currently in ADPro’s image buffer is handled by programs or modules called “operators”.

There are several ways of invoking an operator, depending upon the type of user interface you’re using.

With the List Interface, scroll the **Operators** list until the operator module’s name appears within view, then double-click on the name. Each operator will act differently, so consult Chapter 8 for the specific controls.

With the Button Interface, invoking a previously-set operator button is as simple as clicking once on the button corresponding to the operator or pressing its keyboard equivalent.

With either interface, you can open the floating Operators list and double-click on the module name. Any change (scrolling or selecting) you make to the List Interface **Operators** list automatically appears in the floating **Operators** list, and vice versa.

5.5.2 What Gets Modified?

Most operators modify the raw image data. When it does this, any rendered image data which was available is erased. This is done because the rendered image does not represent the raw image data anymore. To update the rendered image, you must use the **Execute** command (explained in Section 5.6.4).

Some operators actually *create* rendered data or place a modified version of the

raw data in rendered space. In these cases, the rendered data can be displayed with the **Redisplay** (also see Section 5.6.4). Note that, in this case, if you press the **Execute** button, the specially modified version might be overwritten and that the operator would have to be executed again to get it back. The best example of this is the DCTV operator. Try selecting the **Redisplay** and **Execute** buttons after executing this operator and compare the results.

5.6 Displaying an Image

The process of displaying the image involves selecting the render screen and rendering to it. This section will describe the controls available to do so. Although some operator and saver modules handle the display of image data to external graphic devices, this section will focus more on the main display controls.

5.6.1 Render Screen

The render screen is where ADPro displays the image data. It can be on a native Amiga screen or, assuming a display module is available, on an alternate graphics board or device. The term “render screen” is somewhat misleading, because you can even render to a window on any currently open public screen, but it will be used to refer to all types of display “destinations”.

This section will describe how to select a render screen and what types of controls you have over the visible area.

At any time, you can see a description of the currently selected render screen within the Screen Information area (see Figure 5.1 or 5.2 for an example). Information in this scrolling list includes the selected screen mode, the visible area within this screen, the number of colors selected, the dither and dither amount setting, and whether the screen mode, screen depth, and palette are locked.

Set Render Screen

Selecting the **Set Render Screen...** button will display the Set Render Screen requester, to allow you to change the attributes of the render screen. The **Set Render Screen...** menu item under the **Display** menu performs the same function.

This requester is a custom screen mode requester, not unlike the Amiga’s standard screen mode requester available with AmigaOS 2.1 and later. However,

ADPro's screen mode requester will work under AmigaOS 2.04 as well, and will show the available 24-bit graphics display screens (those supported by display modules), and public screens. Consult Chapter 9 for examples of how each type of display appears within the Set Render Screen requester.

The width and height of the *viewable* portion (which is not to be confused with the visible image within the non-overscan area) of the selected screen can be specified in the **Width:** and **Height:** input fields. When a screen mode is first selected, these fields are updated to the selected mode's dimensions. If you make this viewable area smaller than this initial value, the screen will be in the mode you selected, but the image will only be visible within the defined area.

NOTE Some screen modes do not support different screen sizes. They are noted as such.

NOTE Under AmigaOS 2.04 only, the render screen must be at least three-fourths of the monitor width. Even if you entered a smaller screen width, it will show at least this amount of the width.

Also, if you are rendering to an Amiga screen (or to a mode which acts like an Amiga screen, such as the Picasso color-mapped modes), the render screen must be at least 200 pixels high. If you enter a value smaller than this, it will open a screen 200 pixels high. If you are rendering to a window or to a high or true color graphics device, this restriction does not exist.

Overscan Modes

The **Overscan:** cycle gadget can be set to one of the following choices. Each choice will update the **Width:** and **Height:** input fields to the selected overscan's size.

Regular Size	This defines the standard, non-overscan dimensions of the selected screen. This choice may also be labelled as Minimum Size . An example is 320 x 200 for NTSC Low Res.
Text Size	This defines the current Text Size dimensions defined in the Overscan Preferences editor.
Graphics Size	This defines the current Graphics Size dimensions defined in the Overscan Preferences editor.
Maximum Size	This defines the largest screen dimensions that your graphics hardware can <i>safely</i> support.

Extreme Size

This defines the largest screen dimensions that your graphics hardware can support and which a mouse pointer (but no other sprites) are still visible.

You can create a screen of any size between **Minimum Size/Regular Size** and **Maximum Size**. However, you cannot create view sizes between **Maximum Size** and **Extreme Size**. You can enter these sizes into the fields, but only an area of **Maximum Size** (if between these two sizes) will be viewable at once; autoscrolling will be enabled to display the hidden areas.

Changes To Render Screen Dimensions

ADPro's screen controls offer a selection of more than 208 possible video modes, as many or more than any other Amiga program at this time. Once an image has been fully rendered, many of the changes from one screen mode to another are instantaneous.

NOTE Changes to any of the screen controls do not affect the raw or rendered data in any way. Therefore, such changes can be quickly undone.

Given a fully rendered image, changes from low resolution to high resolution will be instantaneous if the color mode chosen is allowed in both low and high resolution.

All color modes are supported in all vertical sizes. Selecting an NTSC vertical size on a PAL machine simply truncates the screen size at the NTSC lower boundary. Selecting a PAL vertical size on an NTSC machine simply extends the screen size below the visible lower border of your NTSC screen.

Given a fully rendered image, changes from one vertical size to another are instantaneous. Aspect correction for changing between interlaced and non-interlaced screens can be accomplished using the digital scaling capability of ADPro.

5.6.2 Number of Colors

The number of colors that will be used for the rendered image is specified by the **Colors:** slider and **Mode:** cycle gadget in the Set Render Screen requester.

The **Mode:** cycle gadget can be set to one of the following (although some options may not be available, depending upon your display hardware, the render screen selected, and the state of the **Available Modes Only** menu item):

Color mapped	This option allows you to select 2, 4, 8, 16, 32, 64, 128, or 256 colors. It will produce rendered image data. If the image buffer contains gray image data, the Mode: cycle gadget will be locked to this mode.
Extra half-brite	This option allows you to render in EHB (Extra Half-Brite) mode, which is 64 colors. It will produce rendered image data.
HAM	This option allows you to render in HAM (Hold And Modify) mode, which is 4,096 colors. It will produce rendered image data. Non-AGA machines cannot display images in this mode in high resolutions—only in low resolution.
HAM 8	This option allows you to render in HAM 8 mode, which is 262,144 colors. An AGA-equipped machine is required to display images in this mode.
Hi color	This option allows you to render in 15 (32K colors) or 16 (64K colors) bit-planes, and is only available with extra display hardware.
True color	This option allows you to render in 24 (16M colors) bit-planes, and is only available with extra display hardware.

The cursor left and right keys allow you to cycle through the available **Colors:** choices for the currently selected render screen.

Modes Supported By Your Hardware

Each render mode has its own number of colors limit. The **Available Modes Only** menu item under the **Settings** menu controls whether you can render in modes and number of colors that are not supported by your Amiga's display hardware. If this menu item is checked, then you will never be able to set the **Colors:** slider to a value which isn't supported on your machine. However, if the menu item is not checked, then you can create images viewable by others with the appropriate display hardware. The best example is when you want to create AGA images (such as high resolution HAM8) on your non-AGA machine; you'd uncheck **Available Modes Only**.

NOTE Note that if you render an image in a mode not supported by your Amiga's hardware, the rendered image data will still be created and can be saved—just not displayable on your machine.

A-RES and A-HAM Restrictions

The A-HAM data format is a variation on Sliced HAM (SHAM) popularized by Rhett Anderson, formerly of Compute Magazine. In A-HAM, a new set of 16 base color registers is chosen for each line in the image. This increases the color fidelity of a HAM image at the expense of increased machine and display overhead.

ADPro will read A-HAM images in the same file format (as registered with CATS, Commodore Applications and Technical Support) used by NewTek's Dynamic HAM image files.

The A-RES modes are 16 color hi-res variants developed by ASDG, Incorporated. The A-RES file format was designed to be backwards compatible with NewTek's Dynamic Hi-Res format as registered with CATS.

A-RES allows all 4,096 colors supported by the Amiga to be present on one hi-res screen at the expense of extreme processor overhead while an A-RES image is being displayed.

NOTE The A-RES modes are not for general purpose use as are the standard Amiga display modes. There are many limitations to what can be done while displaying an A-RES image and not all images lend themselves to rendering in this mode.

The generation of A-RES and A-HAM images is not fully supported. While ADPro will continue to be able to read these images, it can no longer generate them. This decision was reached after careful consideration based upon knowledge of future developments.

5.6.3 Dither Type

Dithering is a technique for achieving greater color fidelity at the expense of spatial fidelity (image sharpness). ADPro supports the following dithering types: **None**, **Floyd**, **Burkes**, **Sierra**, **Jarvis**, **Stucki**, **Random**, **Large Ordered**, and **Small Ordered**.

The **None** setting turns off dithering.

The **Floyd** setting, short for Floyd-Steinberg, is the tightest of the dithers and produces good results for all video modes.

The **Burkes**, **Sierra**, and **Jarvis** settings produce various levels of tightness between **Floyd** and **Stucki**.

The **Stucki** setting is the sparsest (and most time consuming to compute) and produces good results where just a little dithering is desired.

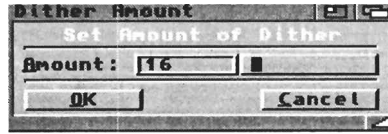


Figure 5.4: Dither Amount control panel.

The **Random** dither is useful in the preparation of successive images which will become part of an animation. While the other dithering techniques produce superior results, they may also produce an undesired flickering when multiple images are animated. The Random dither does not have this problem when animating.

The **Large Ordered** and **Small Ordered** settings are also useful for animations. The large version uses a 16 x 16 dither matrix, while the small version uses a 4 x 4 one.

With the exception of the Random and Ordered dithers, the dithers are presented in the order of greatest to least effect on the image. However, counter to intuition, they range from fastest to slowest.

Dithers 1 (**Floyd**) through 5 (**Stucki**) are each “error diffusion” dithers. The added computation time of the higher numbered dithers results from incorporating a greater number of pixels into each dithering computation. This also accounts for why the higher numbered dithers affect the image less. That is, the error diffusion is spread over a larger number of pixels (thus affecting each pixel less).

Controlling the Amount of Dithering

For the **Random** and **Ordered** dithers you can control how much of an effect they will have on the image. The **Dither Amount (n)...** menu item under the **Display** menu, when selected, displays the Dither Amount control panel (shown in Figure 5.4). You can select a number between 1 and 256. The *n* in the menu item will be updated to the currently selected amount. This item is only available if one of the **Random** or **Ordered** dithers is selected.

If you are using an amount of 16 or less, then you should use the **Small Ordered** dither. However, if your amount is greater than 16, use **Large Ordered**.

Using the Dither Setting

To select a particular dither, choose the appropriate item in the **Dither** sub-menu (under the **Display** menu).

After selecting the dithering style of your choice, and any other modifications to the image using ADPro's other image processing capabilities, select the **Execute** button to render the image.

NOTE Dithering is applied during rendering and has no effect on the raw 8 bit-plane grayscale data or 24 bit-plane color data. Therefore, you may change and rechange the dithering setting at will without affecting the original data.

Dithering gives its best results when displayed in the high resolution video modes. Dithering provides a benefit in the HAM modes but can slightly increase the amount of HAM fringing present in the rendered image. Of the dithering choices available, the **Random** dither is the least likely to introduce HAM fringing but is the weakest dither.

5.6.4 Rendering Controls

The actual controls for displaying an image are handled by the **Execute** and **Redisplay** commands. They are described in this section, along with the render screen scrolling controls.

Execute

Selecting the **Execute** button will cause any changes you might have made in the screen, color, or image controls to be put into effect. The **Execute** menu item under the **Display** menu performs the same function.

If the only change you've made is a change of screen width or height, selecting **Execute** will usually cause an image to render instantaneously.

If the image you are rendering is taller than the currently defined screen, then the image will appear to overwrite itself. This is not actually happening. It is only used to indicate that the image is taller than the viewable screen area. Clicking on the image screen while ADPro is rendering will cause the ADPro control screen to pop to front.

Once the rendering has stopped, you can view hidden areas of your image by using the four cursor keys located on your keyboard. Added control is available by holding down one of the qualifier keys as you press a cursor key. Diagonal scrolling is available with the numeric keypad. Use Table 5.5 as a guide.

Key Sequence	Scroll Amount
←/→ ↑/↓	width/5 or 64 pixels, whichever is larger height/5 or 40 pixels, whichever is larger
Alt + ←/→ Alt + ↑/↓	half of ←/→ distance half of ↑/↓ distance
Shift + ←/→ Shift + ↑/↓	width of displayed image left/right height of displayed image up/down
Ctrl + ←/→ Ctrl + ↑/↓	leftmost/rightmost part of image topmost/bottommost part of image

Table 5.5: Render screen scrolling controls.

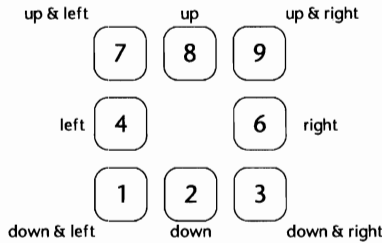


Figure 5.5: Render screen scrolling via numeric keypad.

To switch back to the ADPro control window, either click the left mouse button while the mouse pointer is above the render screen or press the **Esc** key. Alternatively, you can press the **Spacebar** to toggle between the control window and render screen.

NOTE If you click on the render screen, then click in the middle part of the ADPro's front/back gadget, the render screen will reappear. To avoid this "endless loop", click in the upper-right corner of the render screen to access the other open screens.

To cycle through the other open screens, click the left mouse button while the mouse pointer is in the upper-right corner of the render screen. The keyboard sequence of **Left-Amiga + M** will always cycle screens.

NOTE If you have a high color or true color mode selected, the only way that rendered image data (as opposed to the raw image data) can be displayed is if you perform a **Rendered.To.Raw** operation or if rendered data only exists. Otherwise, if both raw and rendered data exist, the raw image data will be displayed.

Also, the color-mapped modes will display EHB, HAM, and HAM-8 choices, even if the display device cannot support it, if the **Available Modes Only** option is not selected.

Redisplay

Selecting the **Redisplay** button will cause the current image data to be displayed on the screen defined by the **Set Render Screen...** setting. This screen can either be a native Amiga screen or even, if a supported graphics board is selected, a 24-bit display. The **Redisplay** menu item under the **Display** menu performs the same function.

If you have made any changes to the color or screen controls and have not selected **Execute**, the image displayed after selecting **Redisplay** will not show the effect of any of these changes.

The same scrolling controls available when pressing **Execute** are available with this command as well.

5.7 Using the Balancing Controls

This and the next section describe the color controls available in ADPro. Each of these controls contributes to or affects the choice of colors which will be used in the rendering calculation.

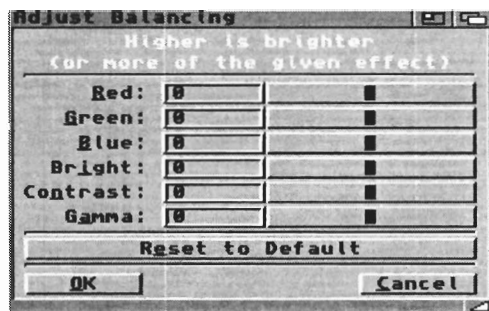


Figure 5.6: The Balancing control panel.

Figure 3.8 depicts how these controls fit in to the flow of data through ADPro.

Notice that the color controls merely filter but do not actually modify the raw data. Therefore, you can always undo a change made to a color control and get back to the original rendered image.

The Balancing control panel (shown in Figure 5.6 and displayed by selecting the **Balancing...** menu item under the **Display** menu) contains the six color adjustments which affect the rendered image.

Each control has its own input field and horizontal slider. These controls will be described in detail later in this section.

At the bottom of the panel you'll find three buttons which will either accept (**OK**), reset (**Reset to Default**), or cancel (**Cancel**) any of the changes you've made to the color controls.

5.7.1 Color Map Basics

Whenever ADPro renders an image, it passes all of the raw color or grayscale data through a series of adjustments before actually using them in the rendering calculation. Before discussing the color balancing features of ADPro, let's first give some background information about how the adjustments are applied.

ADPro stores each pixel of a color image as 3 values (one each for red, green and blue), each of which can range from 0 to 255. Grayscale pixels are stored as a single value which can range from 0 to 255.

Figure 5.7 shows a linear (neutral) color map. A color map is a relationship between input intensities and output intensities. Internally, ADPro maintains a color map for each of the red, green, and blue components of colored data,

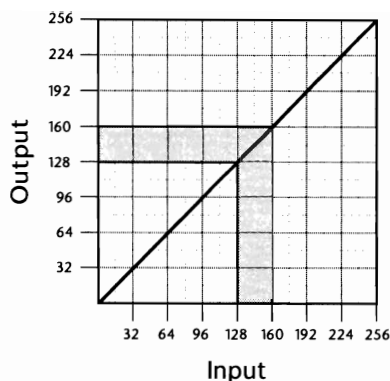


Figure 5.7: A linear (neutral) color map.

and a grayscale map for gray data.

The color map shown in Figure 5.7 is said to be linear, or neutral, because it is a single straight line going from corner to corner in the color map. Notice that an incoming intensity of 128 is output unchanged as 128. The same is true for all intensities from 0 to 255; that is, they are output unchanged.

5.7.2 Red, Green, and Blue Adjustment

ADPro offers control over the individual brightness of the red, green, and blue data. Grayscale brightness is directly controlled by the main brightness control so no additional control is necessary.

Having this individual control allows for simple global color balancing changes. For example, an image which has far too much red can be balanced by decreasing the global brightness of all of the red data in the image.

Note that the individual brightness controls suffer from the same drawback as the main brightness control (described in Section 5.7.3). That is, the more heavily they are used, the more detail will be lost due to the color map being clipped against its minimum and maximum values.

The red, green, and blue adjustments in ADPro all range from -50 to 50, with 0 being the neutral value.

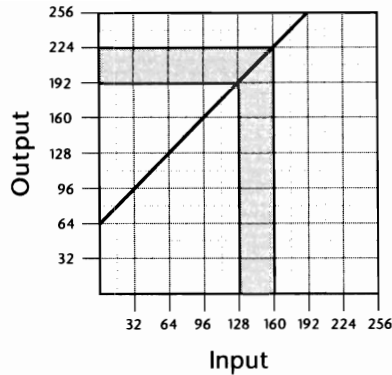


Figure 5.8: A color map showing an increase in brightness.

5.7.3 Brightness Adjustment

The brightness adjustment globally modifies the general brightness of an image. It does this by uniformly shifting the color map upwards or downwards. This is shown in Figure 5.8.

Here, the two input intensities are 128 and 160 (a difference of 32 intensity levels). Notice that they are output as 192 and 224 (with exactly the same difference in intensity levels). Similarly, all input intensities will be shifted upwards (or made brighter) by the color map shown in Figure 5.8.

The brightness adjustment is not without its drawbacks. Notice that an input value of 0 (in the color map shown in Figure 5.8) is output as 64. This means that the darkest intensity in the image will have an intensity of at least 64 which may not be acceptable. Also, note that all values from 192 to 255 all map to the same value—255. This means all details which had intensity levels in that range will become lost.

The brightness control in ADPro ranges from -50 to 50, with 0 being the neutral value. Setting the brightness control to a positive value uniformly shifts the color map upward (towards a brighter image). Similarly, a negative value causes the image to be shifted towards darkness.

5.7.4 Contrast Adjustment

The contrast control globally modifies the contrast of an image. Contrast adjustments can be visualized by thinking of the neutral color map being pivoted around its center point. At one extreme, the color map becomes flat which

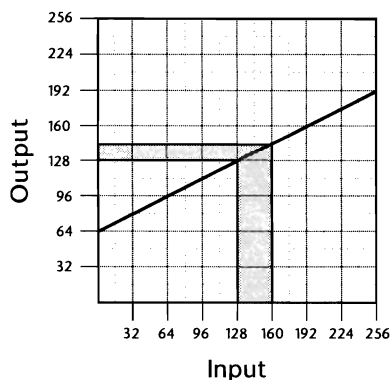


Figure 5.9: A color map showing a decrease in contrast.

means that all input intensities map to the same output intensity (no contrast). The other extreme is a vertical line for a color map. This produces an image with exactly two intensities (maximum contrast).

Figure 5.9 shows how input intensities 128 and 160 (a difference of 32 intensity levels) are mapped to output intensities 128 and 144 (a difference of only 16 intensity levels). Notice that the difference between two input intensities has been reduced. This produces a commensurate decrease in visible contrast.

Notice, again, that contrast loses some amount of visual detail just as the brightness adjustment. Specifically, you note that in the color map shown in Figure 5.9, input intensities which could have ranged from 0 to 255 are now restricted to the range of 64 to 192. This may or may not be acceptable for any given image.

The contrast control in ADPro ranges from -50 to 50, with 0 being the neutral value. Setting the contrast control to a positive value uniformly pivots the color map around its center in counter-clockwise direction (towards the vertical) which increases visible contrast.

5.7.5 Gamma Adjustment

The gamma adjustment provides a way to significantly brighten an image without losing much detail. It does this by introducing a curve into the color map whereby the color map is shifted upwards or downwards (made brighter or darker respectively) but no portion of the color map gets clipped to the maximum or minimum values.

Figure 5.10 shows an example color map which has a positive gamma ad-

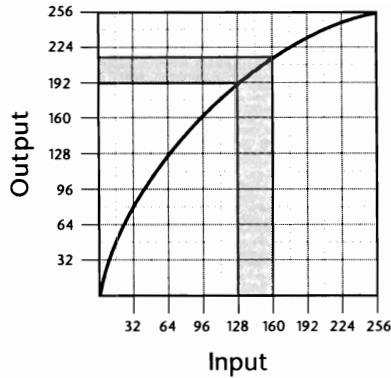


Figure 5.10: A color map showing an example of gamma correction.

justment. Notice that the two input intensities, 128 and 160 are output as intensities 192 and 216. They are, therefore, brighter.

The gamma adjustment also affects the contrast of the image. In the darker part of the spectrum, contrast is increased. However, in the lighter part of the spectrum, contrast is decreased.

The gamma control in ADPro ranges from -50 to 50, where 0 represents no gamma adjustment.

The overall effect of gamma adjustment is usually quite satisfactory and we recommend its liberal use.

5.7.6 Balancing Settings Stored In IFF Files

Whenever raw image data is saved in the IFF format, ADPro includes the current locations of each of the balance settings. The positions of each balance setting will be restored whenever an 8 or 24 bit-plane IFF file is loaded which contains stored settings.

The saving and restoring of the positions of the balance settings are not supported in any other format other than the IFF format. However, the effect of the balance settings will be preserved whenever raw image data is saved, regardless of format.

After loading a raw image (an image which is not color mapped), you may be presented with a panel which states that the "current balance settings do not correspond to actual data" when attempting to enter the Balancing control panel. This means that the image which was loaded did not contain ADPro-

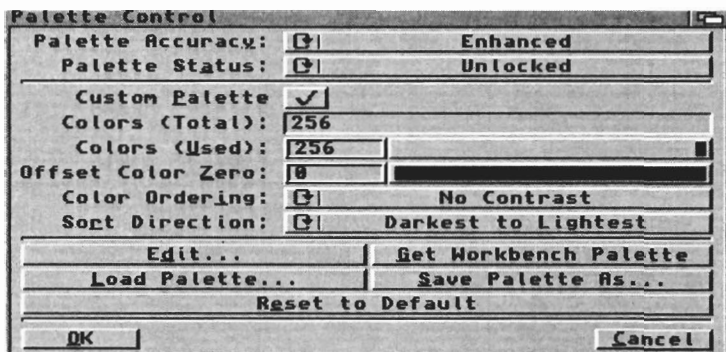


Figure 5.11: The Palette control panel.

specific information about where to place the knobs in the Balancing control panel.

If the loaded data contained a color look-up table, that color look-up table will be loaded and effective until balance settings are “accepted”. So, should you wish to use the color look-up table actually contained in a raw image, do not select the **OK** button in the Balancing control panel. If you should enter the Balancing control panel, select **Cancel** to preserve the color look-up table loaded from the file.

OKing a set of balancing controls has the effect of completely overwriting any color look-up table which might have been loaded with image data.

5.8 Using the Palette Controls

ADPro provides very flexible and powerful palette controls. The Palette control panel is shown in Figure 5.11 and is described in this section. We’d like to mention that because the palette controls are so flexible and powerful, we cannot envision all the different effects and uses they may have. This section will describe the basic operation of the palette controls. We encourage you to experiment with the palette controls to discover new uses.

5.8.1 Palette Editing

Selecting the **Edit...** button will bring up the 256 color palette editor screen.

After editing, the effect of the new palette is immediately viewable by selecting the **Redisplay** button. However, the image which will be displayed will simply



Figure 5.12: The 256 color Palette Editor.

be the previously rendered image with a new set of colors. To rerender the image (that is, go through the entire data flow as shown in Figure 3.8), you can select the **Execute** button.

If you have enough memory, you will enjoy the benefits of being able to edit all 256 colors on-screen at one time. The Palette Editor is accessed via the button marked **Edit...** on the Palette control panel. If you do not have enough memory available to run this 256 color Palette Editor or if this editor cannot be located on disk, an error message will be displayed.

Color Adjustments

The first thing to note is that although the Palette Editor provides access to all 256 colors on screen at one time, it does not circumvent the Amiga's 12 bit palette width. Therefore, colors which are quite close together in 24 bit space will be shown as the same color in the palette editor. For example, the color (128, 128, 128) will be shown as the same color as (127, 128, 129). This is because most Amigas can display only 16 shades of each primary, whereas the editor allows you to edit 256 shades of each primary.

Simply stated, although you can set each color register to one of 16 million colors, colors which are defined with similar values may not be apparent.

Also, pressing the **Return** key in the integer input fields does not advance you to the next integer input field. You can use the **Tab** to move forwards and **Shift + Tab** to move backwards.

The currently selected color (the one whose values are usually shown in the **RGB** and **HSV** sliders and input fields at the bottom of the screen) is shown with a yellow highlight.

You can copy any color from one location to another by selecting the color to be copied, then select the **COPY** button, then select the color to be overwritten. Two colors can be swapped in the same way (instead of hitting the **COPY** button, select the one marked **SWAP**).

Range Selections and Adjustments

The palette editor contains many powerful capabilities for operating on ranges of colors. Colors which are part of a range are shown with either green or red highlights.

The highlights themselves can sometimes alter your perception of the colors they surround. Therefore, you can press the right mouse button at any time to cause all graphical elements on screen to fade to black—leaving only the 256 colors. (In fact, you can use the sliders at the same time the screen is blacked out by first depressing the left mouse button over the slider, then depressing the right mouse button to black out the screen).

You can select a range in two ways. First, you can select the first element of the range, then select the button marked **SELECT RANGE**, then select the last element in the range. All of the colors between the first and last elements will be highlighted in green.

If you hold down the **Alt** key and perform the steps above, the color will be highlighted in red. The difference between red and green highlighting will be explained later.

Range deselection can be accomplished the same way. Select the first element to be deselected, select the **DESEL RANGE** button, then select the last element to be deselected. The first and last elements, and all colors in between, will be deselected.

Ranges which are defined with green or red highlights are said to be **explicit** ranges (as opposed to implicit ranges). Consult the following as a guide to what each highlight color represents:

yellow	current color register
green	register will actively participate in range operations
red	register will passively participate in range operations, acting as “place holders” and not being affected by them

All of the commands with the word **RANGE** in their name operate slightly differently when an explicit range is defined or not. When an explicit range is defined, the command will affect only those colors which are part of the explicit range (highlighted by red or green). When an explicit range is **not** defined,

the commands will affect all colors in between the first element selected and the second element selected.

The best way to learn how these explicit and implicit range operations affect the color palette is to go through a tutorial. See Section 4.3 for a lesson on these operations.

The difference between **RGB RANGE** and **HSV RANGE** is that the **RGB RANGE** command builds a ramp through the RGB color space (while the **HSV RANGE** command uses the HSV space).

The **HSV RANGE** command is very handy for computing ranges in which all colors have the same intensity but have different hues, etc.

The **ADJUST RANGE** command allows you to adjust the color balance of an entire range of colors simultaneously. To use the **ADJUST RANGE**, you must have an explicit range defined. Select which type of adjustment you wish to make, either in RGB space or HSV space and either ADDitive or PCT (percentage), then select the **ADJUST RANGE** button. After you are done adjusting, select the **ADJUST RANGE** button again to exit the adjusting mode.

5.8.2 Custom Palette Adjustments

This section describes when the Palette control panel settings will be used, as well as how to use them.

When Are The Palette Controls Used?

In general, the palette controls are used in the rendering calculation only when the **Custom Palette** checkbox (in the Palette control panel) is checked.

If this control is not “on”, changes to the palette controls will not be honored, except for the palette’s status and depth.

As described in Section 5.8.3, a palette’s status can be **Locked** or **Unlocked**. When **Locked**, the palette is “write-protected”, which means that ADPro’s own color picking technology is disabled. The effect of a **Locked** palette is in force regardless of the **Colors** setting.

As described in Section 5.8.5, when ADPro selects colors, it can do so at two levels of accuracy. For typical applications, the **Normal** accuracy is sufficient. However, some applications, such as creating imagery for display on non-Amiga computers, may benefit from **Enhanced** accuracy mode. The palette depth selection is in force any time ADPro selects a color on its own.

Colors (Total)

The **Colors (Total):** value describes how many colors can occupy the image's palette. This is the same value as what is specified by the Set Render Screen's **Colors:** control—it is displayed here for your convenience.

The value that is displayed here directly affects the **Colors (Used):** and **Offset Color Zero:** values that you can specify.

Colors (Used)

The **Colors (Used):** input field and slider allow you to enter a number ranging from 2 to the total number of colors chosen. Normally, the number of colors used would be set to the maximum number possible for a given screen format. However, there are many situations where it is necessary to render in fewer colors than the screen format will permit.

For example, Amiga based genlocks show full color video through regions of the screen which are set to color 0. If a bitmap were to be rendered including color 0 and then genlocked, bits of genlocked video would poke through the bitmap anywhere a pixel where color 0 would be found.

To create an image which will appear solid when used with a genlock, simply don't use color 0 anywhere in the image. This can be accomplished by instructing ADPro to use one fewer color than is available. Then, specify a value of 1 for **Offset Color Zero:**. This will instruct ADPro to start filling in colors starting at color 1 rather than at 0. The result will be an image which contains $n - 1$ (where n is the total number of colors possible) colors starting at color 1.

As another example, suppose you are required to use a specific 16 color palette and have to render an image with 4 colors (but in 4 rather than 2 bit-planes). This sort of example is a common requirement for animators. To do this, load and lock the required palette. Select a total number of colors of 16 but a number of colors to be used as 4. Specify a value in **Offset Color Zero:** which will offset the colors used to the first of the 4 colors you wish to use. Note that this requires the 4 colors you wish to use to be stored contiguously in the palette.

The number in the **Colors (Used):** input field cannot exceed the total number of colors. Also, the sum of **Colors (Used):** and **Offset Color Zero:** values cannot exceed the total number of colors.

Offset Color Zero

The value in the **Offset Color Zero:** input field is interpreted in different ways at different times. For example:

- When rendering and the palette is **Unlocked**, **Offset Color Zero:** defines where in the palette ADPro will begin choosing new colors. For example, if **Offset Color Zero:** is set to 4, colors 0 through 3 will not be changed and ADPro will start choosing new colors beginning at color 4.
- When rendering and the palette is **Locked**, **Offset Color Zero:** defines where in the palette ADPro begins fetching colors for the rendering calculation. For example, if you wanted to render a 16 color screen with only 15 colors, you can either render with the first 15 colors or the last 15 colors. Setting **Offset Color Zero:** to 0 will render with the first 15 colors while setting **Offset Color Zero:** to 1 will render with the last 15 colors.
- When loading a color palette, **Offset Color Zero:** defines where in the palette ADPro will begin storing the loaded values. For example, to create a 16 color palette from two 8 color palettes, load the first 8 color palette with **Offset Color Zero:** set to 0. Load the second 8 color palette with **Offset Color Zero:** set to 8. Then, lock the palette, and you're ready to render.

The value in **Offset Color Zero:** can be in the range of 0 to the total number of colors minus 2. Also, the sum of **Colors (Used):** and **Offset Color Zero:** cannot exceed the total number of colors.

5.8.3 Palette Contents

This section describes the controls available for modifying the contents of the palette.

Sort Direction

When ADPro chooses colors and places them into the palette, it can sort them by increasing or decreasing brightness. This button is a toggle which controls the sort direction. Of particular note is that color 0 is the color which will be shown as a border around non-overscanned images. In general, you'd like this to be as dark as possible (**Dark to Light**) so as not to be distracting. However, this can be overridden by selecting **Light to Dark**.

Contrast or No Contrast

When ADPro selects colors, it can order them dark to light or light to dark. In doing so, it may place two very similar colors in color registers 0 and 1. This makes viewing and editing of the palette very difficult since most palette requesters use color registers 0 and 1 as their background and highlight pens, respectively.

If you set the **Color Ordering:** cycle gadget to **Contrast**, then ADPro will examine color register 0 and place a contrasting color in register 1. This eliminates the difficulty in using most palette requesters.

When the cycle gadget is set to **No Contrast**, ADPro will order its palette with no special provision for distinguishing color registers 0 and 1.

Load Palette

Selecting the **Load Palette...** button causes a file requester to appear. Using it, you can select a file from which ADPro will attempt to read a palette. All Amiga format images, such as 2, 4, 8, and 16 color images, have color palettes. However, raw image data, such as 18, 21 or 24 bit-plane files, do not. Palettes can also be loaded from IFF “brush” files.

Palette loading is affected by several other factors in the Palette control panel. Specifically, if the number of colors in the palette to be loaded exceeds the number of colors available to be loaded into, the excess colors will be ignored.

For example, if the total number of colors permissible at the time **Load Palette...** is selected (as defined by the value in the **Colors (Used):** input field) is 15, and the palette to be loaded contains 32 colors, only the first 15 will be loaded.

In fact, if the value in **Offset Color Zero:** is non-zero at the time **Load Palette...** is selected, then the number of colors which will be loaded (in the example cited in the previous paragraph) will be lower than 16.

Note that a loaded palette will be overwritten if the palette is not in a **Locked** condition and you select the **Execute** button to render an image.

Save Palette

The currently defined palette can be saved to a file by selecting the **Save Palette...** button. Upon doing so, a file requester will appear. After selecting a filename, the palette will be stored to disk.

Note that the saved file will contain only a palette. It will not contain any image

data. Therefore, use caution when selecting a pre-existing file to store the palette, as that file will be overwritten with palette information. Its previous contents will be lost.

Get Workbench Palette

This command is very useful for creating imagery which will be displayed on the Amiga Workbench. Selecting this command will cause ADPro to load the currently defined Workbench colors into the palette starting at the color defined by **Offset Color Zero**.

This command fetches *all* the colors in the Workbench palette, as opposed to only four in previous versions of ADPro.

Therefore, consider the **Get Workbench Palette** button as a short hand for loading only the first four colors of the Workbench. Should you require access to the palette of a more than four colored Workbench, use the palette loading facility to load a palette from the `ENV:sys/palette.ilbm` palette preferences file or other pre-saved palette file.

5.8.4 Palette Status

The palette status button can be set to one of two values: **Locked** and **Unlocked**. This defines whether or not ADPro will execute its color choosing routines prior to rendering or skip directly to rendering using the pre-existing palette contents.

When the palette is **Unlocked**, ADPro will analyze the raw image data and catalogue various statistics about the colors it contains. It will then optimally choose a set of colors which best approximates the entire image.

When the palette is **Locked**, ADPro will use the prior contents of the palette and will not choose a new one. Each pixel in the image will be rendered in the color chosen from the palette which best matches the intended color.

Referring to Figure 3.8, you can see that color adjustments and dithering still affect the final image even when the palette is **Locked**.

The palette **Locked** condition can also affect image loading. If the palette is in the **Locked** state and the image you wish to load contains a palette, ADPro will ask you if you wish to load the palette from the file or keep the previously defined palette. This will occur whether or not the **Custom Palette** is selected.

5.8.5 Palette Accuracy

When ADPro selects colors, it can do so at two levels of accuracy. Typically, the **Normal** palette accuracy is sufficient for nearly all imagery intended for use on an Amiga based computer.

Other computer systems or display technologies can take advantage of more accurate color palettes. For example, VGA systems can display up to 256 colors simultaneously chosen from an 18 bit-wide palette. Commodore's Hi-Res graphics card can display up to 256 colors chosen from a palette 24 bits wide.

Consider the difference between a palette's depth and its width (accuracy). Palette depth can be thought of as the number of colors which can be simultaneously displayed. In the case of VGA, this is 8 bits deep (256 colors). The width of the palette can be thought of as the precision with which each color can be chosen. In the case of VGA, this is 18 bits wide.

A 32 color image can be displayed on both the Amiga's own screen and on Commodore's A2410 Hi-Res graphics card. For display on the Amiga's own screen, the 32 colors are chosen from a total spectrum of 4,096 choices. For display on Commodore's Hi-Res graphics card, the 32 colors can be chosen from a spectrum of 16.7 million choices.

ADPro's **Enhanced** palette technology carries 24 bit-plane accuracy through all color conversion computations. By offering selectable palette accuracy, ADPro makes it possible to prepare high quality imagery for nearly any computer display hardware including future Amiga display technologies, such as the Advanced Graphics Architecture.

The enhanced palette technology does require some additional memory. If you are in a memory limited environment, you can disable enhanced palette mode operations. If you can spare the memory, we encourage the usage of the enhanced palette mode for all picture generation.

We recommend this mode for any type of processing, except in memory limited environments because it does provide better results in almost all modes and its speed is now acceptable.

5.9 Printing an Image

There are two ways of printing an image out of ADPro, and both are accessed as saver modules.

5.9.1 Preferences-Supported Printers

The PREFPRINTER saver should be used when you want to output your raw image data to a printer that is supported by any standard Amiga printer driver that supports strip printing. As of Workbench 3.0, the following printer drivers do not support strip printing, and as such cannot be used with PREFPRINTER:

- CalComp_ColorMaster
- CalComp_ColorMaster2
- PostScript™
- Tektronix_4693D
- Tektronix_4696

The convenient **Print As...** menu item under the **Project** menu will invoke the PREFPRINTER saver.

NOTE The PREFPRINTER saver also directly supports the FARGO Primera dye-sublimation printer.

5.9.2 Postscript Printers

The POSTSCRIPT saver should be used when you want to output your raw image data to a PostScript™ compatible printer attached to your Amiga or to a PostScript™ file on disk.

5.10 Using the Clipboard

You can save the current raw or rendered image data to the Amiga's Clipboard using the **Edit** menu's **Copy** command. The CLIPBOARD saver is invoked when this menu item is selected, so if both raw and rendered image data is available, the CLIPBOARD saver's control panel will appear for the selection of which type of data to copy to the Clipboard.

The current contents of the Clipboard can be loaded into ADPro by selecting the **Paste** command. The CLIPBOARD loader is used to perform this operation, so your current load setting (compositing and orientation) will be honored.

Note that the Clipboard is, by default, assigned to your RAM disk. Saving large images to the Clipboard will decrease the amount of free memory available to other applications.

5.11 Preferences

This section describes the controls you have over how ADPro looks and functions.

5.11.1 Selecting ADPro's Screen Mode

ADPro's main window and floating module lists can appear on any open public screen or on its own screen (which is also public). The **Set ADPro's Screen...** menu item under the **Settings** menu displays a screen mode requester, from which a public screen may be selected or created.

Opening ADPro on a public screen or as a public screen for other application to open on gives you the flexibility to customize your working environment to be more productive.

5.11.2 Selecting ADPro's Font

If ADPro's window is opened on its own public screen, you can specify the font used for the screen and the window's text. Select the **Set ADPro's Font...** menu item under the **Settings** menu and choose the font to use from the font requester.

NOTE Be careful not to select a wide font that would cause the rightmost menu items not to appear. Otherwise, you might not be able to select the **Set ADPro's Font...** menu item to reset the font to a narrower one.

This menu item is not available when running the window on the Workbench screen. Instead, ADPro's window will use the Workbench's screen font.

5.11.3 Adjusting ADPro's Window Size

If you are running ADPro in List Interface mode, you can use the window's size gadget (in the lower right corner) to resize the window to virtually any size you like. The controls will automatically readjust to the new dimensions. The window's zoom gadget will toggle between the currently defined minimum and maximum window sizes.

In either List or Button Interface mode, you can shrink the window down to just its title bar by toggling the **Minimized** menu item under the **Settings**

menu on. Selecting the menu item again toggles the window back to the previous size.

Minimizing ADPro's window allows you to keep it around on your Workbench (or whatever other public screen), but out of the way. Whenever you need to use the program, simply "maximize" it.

5.11.4 Setting ADPro's Hot-Key Activation Sequence

As a convenience, ADPro installs a Commodities hot-key for you when it's run. This allows you to activate ADPro's window with a simple sequence of key presses. By default, this sequence is **Alt + Shift + a**, but this can be changed using the **CX_POPKEY** tool type.

See Section 13.1 for more information on this tool type.

5.11.5 Customizing ADPro's AppIcon Imagery

You can change the imagery used for the ADPro AppIcon by placing an icon file named "**def_ADProApp.info**" into the "**ENVARC:ADPro**" and "**ENV:ADPro**" directories. If ADPro cannot find an icon in **ENV:ADPro** (the contents of the **ENVARC:** directory get copied to the **ENV:** directory when your computers starts up), it will use the default AppIcon imagery.

5.11.6 Loading and Saving Custom Settings

The **Load Settings...** menu item under the **Settings** menu allows you to load and use a previously-saved settings file.

The **Save Settings** and **Save Settings As...** menu items store the current program settings under the current settings name or under a user-specified name, respectively.

The ability to load and save settings allows you to tailor ADPro's look and parameters to different projects and different needs.

Chapter 6

Loaders

This chapter provides detailed descriptions of the use and operation of all loaders which are included with ADPro. ARexx control of these loaders is described in Chapter 12.

6.1 UNIVERSAL

The UNIVERSAL loader attempts to identify which file format a specific file is written in. If successful, the appropriate loader will be called automatically. The UNIVERSAL loader makes loading images of different file formats a bit more convenient since you do not have to manually switch between loaders. The penalty for this convenience is a minimal slow down due to bringing two loaders in from disk instead of one. But, with a reasonable hard disk drive, this slow down is inconsequential.

The UNIVERSAL loader attempts to find conclusive evidence identifying which file format a given file is written in. However, some file formats do not provide any way of locating conclusive proof of their identity. If the UNIVERSAL loader cannot locate conclusive proof, it does contain heuristics which enable it to guess (correctly) at a file type most of the time.

Some file formats, for example the SCULPT format, cannot be loaded from the UNIVERSAL loader at all.

The UNIVERSAL loader identifies file formats in a two step process. First, it uses file name extensions as a hint of which file formats to investigate first. Second, it checks the internal structure of the file against known facts about each file format. Table 6.1 lists some of the formats detectable by the UNI-

VERSAL loader including the file name extensions which it will recognize as hints.

If the UNIVERSAL loader cannot determine the file format for a given file, it will tell you. The UNIVERSAL loader will also inform you if it can determine which format a given file is written in, but you do not have a loader for that type.

Some of these loaders are available as standard modules in the ADPro package, while others are from ADPro's companion packages—the Professional Conversion Pack (PCP) or the CGM Loader (CGM).

6.2 ALPHA

The ALPHA loader is an IFF-ILBM loader that allows you to use a grayscale IFF-ILBM image as an alpha channel, controlling the transparency of the image to be loaded and composited over the current image in ADPro's image buffer.

Currently, the file specified as the alpha channel must be a grayscale IFF and must have the same width and height as the image being loaded. In this alpha channel file, the whiter the pixel the more the foreground (loaded image's) pixel will replace the background (current image's) pixel, and the darker the pixel the less it will replace it. You can create alpha channel files in any paint program for use as sophisticated masks or artistic effects.

NOTE We highly suggest you use the more general and powerful alpha channel support (explained in Section 5.3.3) instead of the ALPHA loader. Doing so allows you to use non-IFF-ILBM foreground and alpha channel images. The ALPHA loader is available for backward compatibility with ARexx scripts, as well as an quick way of compositing with IFF-ILBM images.

To use the ALPHA loader from ADPro, follow the steps below:

1. If the **Composition** setting (under the **Loaders** menu) is set to **Compose**, switch it to **Replace**.
2. Load the background image, if it isn't already loaded, into ADPro's image buffer.
3. Switch the **Composition** setting to **Compose**.
4. Double-click on the ALPHA loader from the **Loaders** list. This step launches the ALPHA loader.

Format	Extension	ADPro	MorphPlus	Other
ALIAS	.alias, .als			PCP
ANIM	.anim	X	X	
BMP	.bmp	X		
CDXL	.cdxl, .xl	X		
CGM	.cgm			CGM
DEEP	Not used			Third-Party
FLC	.fli, .flc	X		
FRAMESTORE	.fs	X		
GIF	.gif	X		
HAME	.hame	X		
ICO	.ico	X		
ICON	.info	X		
IFF	.brush, .ham,	X	X	
	.iff, .ilbm,	X	X	
	.lbm, .sham	X	X	
IMPULSE	.rgbn, .rgb8	X		
JPEG	.jpg, .jpeg,	X	X	
	.jff, .jfif	X	X	
MACPAINT	.mac, .mpnt,	X		
	.pntg	X		
PAR_PEG	Not used	X		
PCX	.pcx	X		
PICT	.pict			PCP
QRT	.qrt, .dbw	X		
RENDITION	.6rn			PCP
SGI	.bw, .rgb,			PCP
	.sgi			PCP
SUNRASTER	.sun			PCP
TARGA	.tga			PCP
TIFF	.tiff, .tif			PCP
WAVEFRONT	.rla, .rlb			PCP
X	.xwd			PCP
YUVN	.yuvn	X		

Table 6.1: A list of some of the file formats detectable by the UNIVERSAL loader, the filename extensions recognized as hints, and indication of which modules come standard with a particular package.

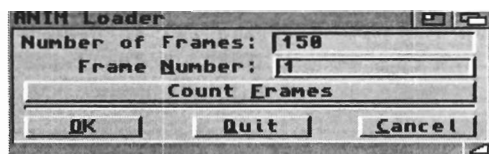


Figure 6.1: The ANIM loader control panel.

5. From the file requester that is displayed, select the image to be composited (known also as the foreground image). This image must be in IFF-ILBM format. Notice that the file requester's title bar displays the type of file being requested.
6. The Image Compositing control screen and panel will appear. Press the **Compose** button to proceed with this operation or **Cancel** to abort it. Additionally, you can press **Replace** to replace the background image with the selected image, thereby cancelling the operation. You can also change the offset and mix values if you wish.
7. Another file requester will appear from which you should select the alpha channel file. Notice once again how the file requester's title bar displays what is needed.

Be sure that the file you select is a grayscale image (in IFF-ILBM format) with the same width and height as the composited (foreground) image, or else the mix levels defined in the Compositing control panel will be used (instead of those defined by the alpha channel file).

After you select a proper file, the foreground image will be composited over the background image.

Be sure you do not delete the **.ALPHA** and **ALPHA** files from the **Loaders2** directory. The ALPHA loader needs both of these files to operate. These files are placed in this directory by the installation program.

6.3 ANIM

The ANIM loader allows you to load any frame within a properly defined IFF-ANIM (Op Mode 5) animation file.

NOTE This version of the ANIM loader supports the new ANIM OpCode-8 (ANIM8) format, which allows for word and long word compression, in addition to the byte compression method used in ANIM5).

After selecting the ANIM loader, a file requester will appear for you to select an ANIM file. Once a valid IFF-ANIM file has been selected, the ANIM loader control panel will appear, as shown in Figure 6.1. The loader will then ask you for the specific frame number (shown in the **Frame Number:** input field) to load. Assuming that the animation has that frame number, it will load that frame into ADPro.

The next time you load an ANIM frame, the number specified in the **Frame Number:** input field will be incremented, assuming you will want to load the next frame in the animation.

Just above the **Frame Number:** input field is the **Number of Frames:** indicator. It will display **0** until you press the **Count Frames** button, which will count how many frames are in the ANIM file and display this value.

A time-saving feature of the ANIM loader is that it remembers the position of the last frame you selected to load. This allows you to load successive frames without having to wait for the loader to search for that frame. For animations made up of a few frames, this feature might not be as evident as with a “multi-frame” animation.

To exit the loader without actually loading a frame, select **Cancel**. The control panel will disappear, returning control back to the main window. If you want to use the time-saving feature just described, select this button to exit the control panel.

If you want to exit the control panel without saving the information about the last frame loaded, select the **Quit** button. **Quit** will close the ANIM file, allowing you to use it in other programs.

NOTE If you want to play the animation with an ANIM viewer, you **MUST** select the **Quit** button after finishing all of your work (or exit the ADPro program). If you fail to do this step, the ANIM viewer will not be able to read the file.

6.4 BACKDROP

Don't let the name, BACKDROP, fool you. The BACKDROP loader is one of the most important loaders provided with ADPro. Using it, you can create a “canvas” upon which you can add other images using ADPro's image compositing features.

At the same time the BACKDROP loader creates an empty bit-map of some given size, you can ask it to fill the bit-map with a specific color or even a gradation of colors. Gradated background fills are very common in presentation and business graphics.

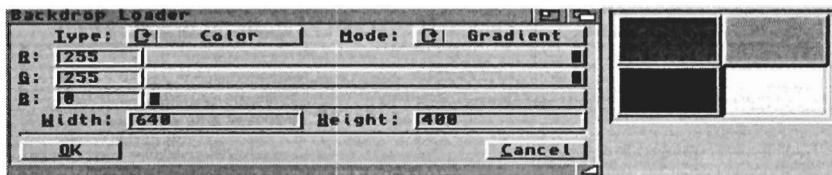


Figure 6.2: The BACKDROP loader control panel.

Figure 6.2 shows the user interface for the BACKDROP loader.

The BACKDROP loader can prepare one of two different types of bit-maps. A 24 bit-plane raw color bit-map can be prepared by setting the **Type:** gadget to **Color**, which is shown in Figure 6.2. Selecting this gadget causes it to toggle to its alternate state, **Gray**, which instructs the BACKDROP loader to prepare a 8 bit-plane gray scale bit-map.

The size of the bit-map to be prepared is specified in the input fields to the right of the markings **Width:** (for bit-map width) and **Height:** (for bit-map height).

The **Mode:** cycle gadget determines how the resulting bit-map will be filled. If in the **Gradient** setting as in Figure 6.2, the bit-map will be filled with a smooth shaded gradated fill. If set to **Fill** the bit-map will be set to one constant color.

You may specify the nature of a gradated fill by selecting each of the four blank rectangular buttons to the right of the control panel. Each of these buttons represents the corresponding corner of the image. That is, the upper left hand button corresponds to the upper left hand corner of the image. To select a square, either click on it with the mouse or use the **Tab** key.

These buttons are enabled whenever you have a gradated fill selected. If you were to toggle the **Mode:** gadget to **Fill**, these buttons become disabled.

A non-gradated color fill can be specified simply by setting the desired color into the red, green, and blue sliders or input fields. When a grayscale bit-map is selected, the red and blue sliders/fields disappear leaving only a slider/field pair marked **G:** (for gray).

As you change the red, green, or blue (or just the gray) component of the color being entered, the numerical value of the color will be updated. Remember that you are entering color components with 24 bit accuracy. The selected color is only accurate to the 12 bit limit of the Amiga's palette. Therefore, slight adjustments in the numerical value of the color may not be visible on-screen.

When ADPro constructs a gradated fill, the color of each pixel is computed as a linear interpolation both horizontally and vertically.

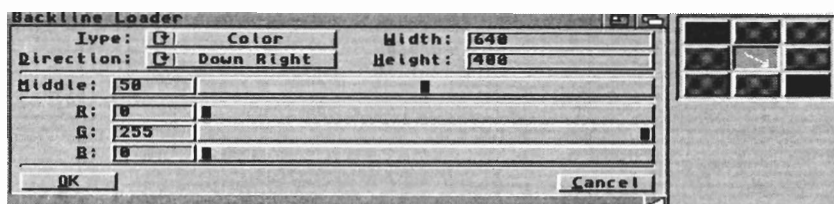


Figure 6.3: The BACKLINE loader control panel.

To copy the color of one corner to another, select the corner you wish to copy from, hold down the **Shift** key, and, finally, select the corner you wish to copy to.

To exchange the position of one color with another, select one color, hold down the **Alt** key, and, finally, select the other color.

6.5 BACKLINE

The BACKLINE loader is similar to the BACKDROP loader in that it is used to create backdrops. BACKLINE differs from BACKDROP in the kinds of gradated background that it can produce. BACKDROP allows you to specify the color of the four corners of the backdrop. This allows for nice vertical or horizontal sweeps. With creative use, the BACKDROP loader can even be coerced into producing diagonal fills.

The BACKLINE loader allows you to specify three colors which will be used to create the fill. BACKLINE creates a smoothly shaded ramp from the first color (the Start color) to the second color (the Middle color), then it smoothly shades from the Middle color to the third color (the End color). The ramp itself can be oriented in any one of eight directions (horizontally, vertically, or along the diagonals).

The shading of the ramp is designed to make it appear like a curved surface. This is done by using the Start, Middle, and End colors to define a mathematical entity called a “spline.”

Figure 6.3 shows the control panel for the BACKLINE loader. You can control the direction of the ramp by setting the **Direction:** cycle gadget to one of eight directions. The color of the currently selected square (its border is recessed) is described by the **R:** (red), **G:** (green), and **B:** (blue) sliders and input fields. To select a square, either click on it with the mouse or use the **Tab** key.

The middle color square (in Figure 6.3) contains an arrow which will always correspond to the ramp direction you selected, and is an additional form of

feedback confirming your selection.

The **Middle:** slider and input field value allow you to specify how (spatially) close the Middle color will be to the Start or End. The value you can specify here ranges from 0 to 100. A value of 0 means that the color in the middle square will coincide with the Start color, while a value of 50 means that the Middle color will lie halfway between the Start and End color, and a value of 100 means that the Middle color will coincide with the End color.

Note that if you specify a diagonal ramp with a **Middle:** color position of 50 percent, the Middle color will, in fact, lie along the diagonal (from corner to corner) only if the area you are creating is truly square.

To copy a color from one square to another, select the color you wish to duplicate, hold down the **Shift** key, and, finally, select the destination.

To exchange the position of one color with another, select the color to exchange, hold down the **Alt** key and, finally, select the other color.

The **Type:** cycle gadget allows you to specify whether the BACKLINE loader should create a grayscale or color backdrop. The input fields marked **Width:** and **Height:** allow you to specify the dimensions of the backdrop.

Selecting the **OK** button causes a backdrop to be created according to the values you specified. The **Cancel** button exits the loader without causing a backdrop to be created.

6.6 BMP

The BMP format is the image file standard under Microsoft Windows 3.0 (and higher), much like IFF is the standard on the Amiga. BMP files come in 1, 4, 8 and 24 bit-plane varieties. While BMP files may be compressed using RLE (run-length encoding), we have not found an application which takes advantage of this capability. Further, the design of BMP's compression with respect to 24 bit-plane images virtually assures that compressed images will be larger than the original versions.

ADPro's BMP loader will read both compressed and uncompressed BMP files in all four depths.

6.7 CLIPBOARD

The CLIPBOARD loader allows ADPro to read pictures from the Amiga's Clipboard device. The Clipboard is a resource which can be shared between

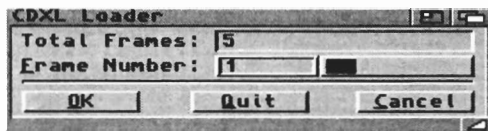


Figure 6.4: The CDXL loader control panel.

applications which support it. Note that text as well as images can be placed into the Clipboard. The CLIPBOARD loader can make use of only picture type information.

Note that the Amiga's Clipboard device is not heavily supported, and those applications that do support the Clipboard may not be expecting any other type of image data to be in the Clipboard except standard Amiga displayable images. Also note that the Clipboard device is often assigned to **RAM:**. This means that any image data written to the Clipboard will consume memory.

6.8 CDXL

The CDXL loader can load most of the image types that the CDXL format supports. Specifically, it supports 1 through 8 bit rendered data, EHB, HAM and HAM8. It also supports 8 bit gray and 24 bit color chunky image data. It allows you to load (preserve) audio and pad data from existing CDXL files which include such information; this information will not be discussed in this manual as it is discussed in the documentation given to CDTV/CD32 developers.

NOTE Although CDTV/CD32 developers will have the most use for these modules, non-developers can still use the CDXL modules as another multi-image single-file solution. Therefore, the use to non-developers is tied to CDXL files created with the CDXL saver.

After selecting the CDXL file to load, the loader's control panel (shown in Figure 6.4) will appear.

The total number of frames in the CDXL file is displayed in the **Total Frames:** field. Specify the frame to load with the **Frame Number:** setting. Select the **OK** button to initiate the load.

The CDXL loader remembers the position in the current file, so that if you invoke the loader again it will be positioned to load the next frame. You can override this by just changing the value in the **Frame Number:** field.

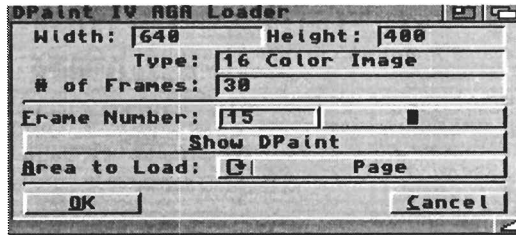


Figure 6.5: The DPaint IV AGA loader control panel.

Select the **OK** button to load the specified frame. When you are done working with a file, exit out of the control panel by selecting the **Quit** button. Otherwise, select the **Cancel** button to preserve the position within the file.

6.9 DPAINT

The DPAINT loader (shown in Figure 6.5) allows you to grab any primary image, swap buffer image, or ANIM frame from a currently running copy of Electronic Arts' Deluxe Paint IV AGA paint program.

NOTE While this loader's panel is displayed, you **cannot** perform any function within DPaint.

The **Width:** and **Height:** text fields describe the area of the current image buffer to be loaded.

The **Type:** text field displays the format of the image data in DPaint's buffer.

The **# of Frames:** text field displays one more than the total number of ANIM frames. This extra frame (frame 0) is for DPaint's swap buffer. DPaint's primary buffer (what you normally work with when not creating an ANIM) is frame 1. If you are working with an ANIM file, the ANIM uses frames 1 through n , where n is the number of frames in the ANIM. This means that there is no difference between a single image and a one frame ANIM.

The **Frame Number:** slider and input field allow you to select the frame to load.

The **Show DPaint** button, when pressed, will display the DPaint screen and currently selected frame. You can press the left mouse button to return to the DPaint loader control panel. This is very useful when working work an ANIM or swap buffer.

The **Area To Load:** cycle gadget selects the image buffer area to import into ADPro. If set to **Page**, the area defined in DPaint's Page Size requester (i.e., the entire image area) will be loaded. If set to **Displayed**, only what can displayed in one "screen-full" will be loaded. Note that DPaint's title bar and toolbox are taken into account when defining **Displayed** area. This means that the image to be loaded is the currently visible area in DPaint. Toggling between these values changes the displayed **Width:** and **Height:** values.

Press the **OK** button to perform a load, or **Cancel** to abort the operation.

6.10 DPIIE

The DPIIE format was defined by Electronic Arts for use in their product "Deluxe Paint II Enhanced" for the IBM P.C. DPIIE images contain 256 colors. Deluxe Paint II Enhanced for the P.C. saves images with fewer than 256 colors in the standard Amiga IFF format.

NOTE If you own or use files from Deluxe Paint IIe on the MS-DOS platform, please note that the DPIIE loader has been removed from this version of ADPro and its functionality absorbed within the IFF loader. The installation script will delete the DPIIE loader from your system, if it exists. The UNIVERSAL loader has also been modified to call the IFF loader if a DPIIE file is selected, so if you use the UNIVERSAL loader to import DPIIE files you will not notice anything different.

ARexx scripts that use the DPIIE loader will have to be modified to use IFF instead.

6.11 DV21

The DV21 loader can process the 21 bit-plane files saved by NewTek's Digi-View 3.0 **only**. A 21 bit-plane file saved by Digi-View 4.0 must be read by the IFF loader.

Files loaded by this module are always considered to be color images. As the image is loaded, it will be padded to a full 24 bit-planes.

To load a 21 bit-plane file saved by Digi-View 3.0, select the DV21 loader and choose the DV21 file from the file requester.

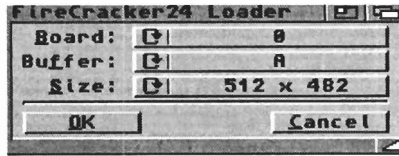


Figure 6.6: The FC24 loader control panel.

6.12 FC24

The FC24 loader (shown in Figure 6.6) allows you to grab the image currently in a Firecracker 24 display buffer.

The **Board:** cycle gadget allows you to select the Firecracker 24 board from which to load. Board numbers start at 0. If you only have one board installed, then this gadget will be ghosted.

The **Buffer:** cycle gadget allows you to select the display buffer (**A** or **B**) from which to load.

The **Size:** cycle gadget allows you to select from one of the board widths: 384, 512, 768, and 1024.

Select the **OK** button to import the currently defined buffer, or **Cancel** to abort the load operation.

6.13 FLC

NOTE The FLC loader was developed by Steve Tibbett, and not ASDG, Incorporated. As such, you should contact Steve Tibbett's BBS at (613) 731-3419 for technical support. This module is included in the ADPro package under agreement with Steve Tibbett.

The FLC loader allows you to import any frame from an FLI/FLC (also termed "flick") file, as generated by programs such as Autodesk, Inc.'s 3D Studio and Animator Pro.

Upon launching the loader, a file requester will appear from which an FLI or FLC file can be selected. Then, the FLC control panel (as shown in Figure 6.7) will appear, showing the name of the file you selected, the resolution of the FLC frames, the number of frames, and the current frame to load (which you can change).

Select the **OK** button to load the specified frame, or **Cancel** to abort the

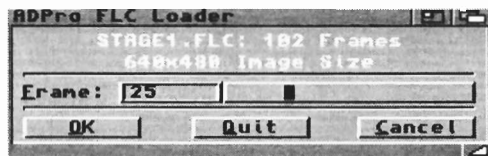


Figure 6.7: The FLC loader control panel.

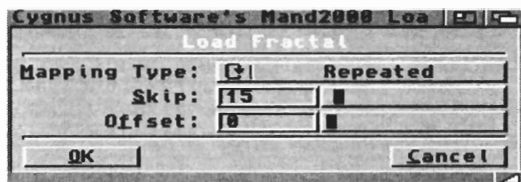


Figure 6.8: The FRACTAL2000 loader control panel.

load. With both these operations, the file is kept open so that the next time the loader is run, it will be at the same frame position. If a frame was loaded, the **Frame:** indicator will advance to the next frame in the file (or wrap back to the first frame if already at the end).

If you have completed your work with the file, selecting the **Quit** button will allow other applications to access the file. No image is loaded when **Quit** is selected.

6.14 FRACTAL2000

Cygnus Software's FRACTAL2000 is a fractal loader for ADPro which converts Mand2000 fractal images into 24 bit colour.

Cygnus Software's Mand2000 can create images with up to 256 colours—the maximum the Amiga's graphics routines support. However Mand2000 can save out the iteration counts for each pixel in a special IFF chunk which FRACTAL2000 loads into ADPro as a 24 bit image. This lets you easily create fractal images with thousands of colours—even on a non-AGA Amiga.

6.14.1 Using the FRACTAL2000 Loader

Some example files have been provided to illustrate how the fractal loader works. These files must be loaded with the FRACTAL2000 loader—not with the UNIVERSAL loader, because the UNIVERSAL loader will identify the files

as being normal IFF images and will try to use the IFF loader to load them. When you load one of these files with the FRACTAL2000 loader, a requester (shown in Figure 6.8) will appear. This requester is identical in behaviour to the colour mapping requester in Mand2000.

The **Mapping Type:** cycle gadget has the following values:

Repeated

In this mode the available colours from the original image are assigned in sequence to iteration counts in the image. The **Skip:** count indicates how many times (i.e., over how many iterations) each colour is used before moving to the next colour. When the final colour is reached, the loader returns to the first colour to repeat the process. Thus, in Mand2000, when the **Skip:** count is greater than one, the colours are repeated that many times. This produces wide bands of each colour. In FRACTAL2000 however, a smooth 24-bit spread is done between successive bands. If the original image was 32 colours, and the **Skip:** count is 10, then the image loaded into ADPro will have 320 colours spread between the 32 base colours.

For best results use a **Skip:** count of at least ten. If the **Skip:** count is set high enough this will produce a separate colour for each iteration level.

NOTE If the **Skip:** count is set to one, then repeat mode produces results identical with using it in Mand2000. Therefore you should always set the skip count higher than one.

Spread, Front spread, and Back spread

In these three modes the loader goes through the colours from the original image just once. In Mand2000, the individual colours are repeated however many times are necessary to fill out the number of iterations. In FRACTAL2000, instead of repeating the colours, a smooth 24-bit spread is done between successive colours. These modes always produce a separate colour for each iteration level. In the **Spread** mode, the colours are spread out evenly. In the **Front spread** mode, most of the colours are concentrated at low iteration levels. In the **Back spread** mode, most of the colours are used at high iteration levels. The **Skip:** count is ignored in these modes.

Mono

In this mode the picture alternates between black and white. The **Skip:** count specifies how wide the bands of black and white should be. In Mand2000, these wide bands are solid white and solid black. In FRACTAL2000, the bands change smoothly from black to white through shades of grey. Higher **Skip:** counts cause

smoother, slower, transitions. A **Skip:** count of one produces results identical with using this mode in Mand2000.

Filled mono

In this mode the picture does a single smooth spread from pure white at an iteration count of zero to pure black at an iteration count of infinity. If your image has a very high iteration count then much detail will be lost in this mode.

In all of the modes except **Filled mono** the **Offset:** slider specifies which of the colours from the original image to start with.

6.14.2 Using the Loader With Mand2000

If you have Mand2000, then the best way to get good results is to experiment with the colour mapping and palette requesters in Mand2000. The 24-bit results will always look roughly similar to the results in Mand2000—just much, much smoother. However the techniques used to get good 24-bit images are quite different from the techniques used for 32 or 256 colour images. Normally you do smooth spreads in the colour requester, using the **Spread** gadget. But, since FRACTAL2000 does smooth spreads for you there is no need for that. Instead, you typically set the **Skip:** count to a relatively high number and set adjacent palette entries to substantially different colours. This can look unimpressive in Mand2000 but when FRACTAL2000 fills in all the subtle shading—it's beautiful!

NOTE To save space on the ADPro distribution disk the sample images were packed with a new and improved compression method. If you try to load these images into Mand2000 you will probably get an 'iteration data corrupt' message. Don't worry. All of the other vital information will have loaded properly, and Mand2000 will recalculate the iteration data for you automatically. Future versions of Mand2000 will, of course, support this improved packing.

NOTE In FRACTAL2000 each separate iteration level will still be one solid colour. Therefore, the maximum number of colours you can get with the loader is limited to the maximum number of iterations the fractal picture has.

One interesting way to get good results without too much work is to load in a fairly standard fractal palette, with smooth changes from colour to colour, and then alternate these colours with black. Then when you set the **Skip:** to a large number, your fractal will smoothly spread from bright colours to black and back again. An easy way to create palettes where alternate colours are black, is by using **Shift-Spread**. The online help for the Mand2000 palette requester gives more information.

Those poor souls who haven't yet purchased the release version of Mand2000 will be unable to adjust the palette or calculate new iteration files. However, you can still get a considerable amount of control using the colour mapping settings in FRACTAL2000. If you have the demo version of Mand2000, you can load the sample iteration files and experiment with different settings in Mand2000, with more immediate feedback. However you will not be able to save your changes.

To create iteration files in Mand2000, find an interesting area and set your palette and colour mapping settings, then select **Save as....** When the requester comes up you will notice an extra requester at the top called **Save options**. Select either **Save Iterations and Location** (which will save the iteration information but no graphics information) or **Save everything** (which will save the iteration information and graphics information). The additional graphics information is ignored by FRACTAL2000, but it can be loaded by ADPro as a regular image file. After saving the image, load it into ADPro with FRACTAL2000. Your colour mapping settings will show up in the FRACTAL2000 requester.

6.14.3 About Mand2000

Mand2000 is a unique fractal exploration program that the Amiga community is raving about.

Amiga Plus says, "Using Mand2000, nothing in the mandelbrot-area is like it was before."

Amiga Magazin described it as, "Pure interactive fractal fun."

ViewPort said, "along comes Mand2000 to steal my weekend, put a 'no vacancy' sign on my hard drive, and etch swirling fantasies on my sleepless eyeballs."

Mand2000's animated zoom and multi-pass drawing means that even slow Amiga's can zoom around the Mandelbrot set and other fractals practically in real time. What other fractal exploration program offers an option for driving around with a joystick? Mand2000 has a multi-window interface with unlimited numbers of fractal windows and multi-tasking requesters. Mand2000 also offers the unique, graphical "show location" function, saving of iteration data, a powerful ARexx interface with many supplied scripts, plus the ability to create many different types of fractal animations.

If this sounds interesting get the demo version, available on bulletin boards and club disks all over the world. Or better still, send us US \$34.95 and we'll mail off the release version immediately. Sorry, no credit cards.

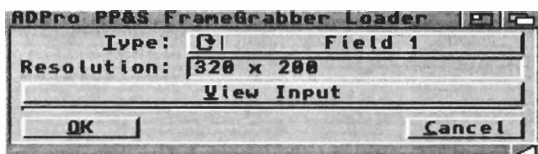


Figure 6.9: The FRAMEGRABBER loader control panel.

Cygnus Software
 33 University Square, #199
 Madison, WI 53715
 USA

European residents also have the option of ordering from our European distributor, ICP-Verlag. The price from ICP-Verlag is 69 DM, plus 5 DM for shipping in Germany, 15 DM for shipping to the rest of Europe. ICP-Verlag's address is:

ICP-Verlag
 Innere Cramer Klette Str. 6,
 D-90403 Nurnberg,
 Germany

If you have any questions, please contact us at the above address or at our e-mail address:

CygnusSoft@cup.portal.com

NOTE Mand2000 and FRACTAL2000 are Cygnus Software productions. As such, you should contact Cygnus Software for technical support questions, and not ASDG.

6.15 FRAMEGRABBER

The FRAMEGRABBER loader allows direct control over Progressive Peripherals, Inc.'s FrameGrabber (a video digitizer). The FrameGrabber can digitize a low resolution interlaced image in 12 bit color in a single video frame time ($\frac{1}{30}$ of a second).

The FRAMEGRABBER control panel is shown in Figure 6.9. Selecting the **Cancel** button will exit the loader without digitizing an image. Any image previously loaded in memory will be unaffected. Selecting the button marked **Digitize** will cause a full video frame to be digitized at that moment.

If you have routed your Amiga video through the FrameGrabber (consult the documentation which comes with the FrameGrabber), selecting the button marked **View Input** will cause the Amiga display to be replaced with the live video coming through the FrameGrabber. All input and processing on the Amiga will be suspended until you click the left mouse button or press the **Tab** key or **Space Bar**. You may also depress the **c** key to cause the currently displayed video to be captured.

When the control panel is displayed, depressing the **Tab** key or **Space Bar** will toggle between your Amiga display and live video. This is equivalent to selecting the button marked **View Input**. As mentioned above, while displaying live video, all input and processing on your Amiga will be suspended until you press the left mouse button, or press the **c** key, the **Tab** key, or **Space Bar**.

The **c** key will also cause a grab to take place when the control panel is displayed. This is provided for consistency with the FrameGrabber's own software.

The image may be grabbed in one of four ways based upon the intrinsic capabilities of the FrameGrabber. The FrameGrabber physically grabs two 320 by 200 fields (each in $\frac{1}{60}$ of a second) in 12 bit color. You can choose to download either field alone, resulting in a 320 by 200 image. Or, you can download both fields stacked one upon the other, resulting in a 320 by 400 image. Or, finally, you can download both fields interlaced together, also resulting in a 320 by 400 pixel image.

If you are digitizing a still scene, we suggest using the interlaced field mode (320 by 400 pixels) as it produces the image data which yields the most exact reproduction of which the FrameGrabber is capable.

If you are digitizing a moving scene, then you may wish to download only one field rather than both. This is because a single field is digitized in a 60th of a second and can stop motion better than two fields together (which may contain some motion blur). In fact, if you select the stacked field mode, then both fields will be downloaded such that you can choose which field to save. In the stacked mode, a 320 by 400 pixel image is produced from two 320 by 200 pixel images stacked one above the other. Using one of the crop operators, you can pick which field contains the better looking data.

NOTE The FRAMEGRABBER loader does not operate correctly on PAL machines.

6.16 FRAMESTORE

The FRAMESTORE loader allows you to import any Video Toaster 2.0 (or later) Framestore image—even compressed Framestore files—directly into ADPro's image buffer in full color; it will **NOT** load Toaster 1.0 Framestores. No Toaster hardware need be present nor active for this operation to work.

Upon invoking the loader, a file requester will appear asking for you to select the Framestore file to load. Once selected, it will attempt to load the file. The meter window will appear, showing the progress of the operation. Once the meter indicator completes its pass, you can operate on or save the image data as you would any other type of image data imported into the ADPro program.

6.17 GIF

The GIF format was defined by CompuServe Information Services, Inc. for storing images in a compressed form on their information network. The GIF format is very widely used on IBM and Apple computers. GIF images may contain from 1 to 256 colors.

The ADPro GIF loader supports both the 87a and 89a GIF standards, including interlaced GIF images. It, however, does not support multi-image GIF files—only one image will be loaded.

To read an image in the GIF format, select the GIF loader and choose the GIF file to load from the file requester.

6.18 HAME

The HAME loader will read files formatted for use with the HAM-E display enhancer from Black Belt Systems. This file format encodes enhanced color information in a standard 16 color high resolution IFF file. The loader decodes this formatting information and converts the loaded image back into true color data.

The HAM-E format places one or more “cookie” lines in with the image data to control how the data is to be interpreted. These lines are stripped from the image as they are read. Therefore, it is normal and expected to load in an image and receive fewer lines than you were expecting.

Note that you can load HAM-E formatted images using either IFF or HAME loaders. If you use the IFF loader, then you can directly view the loaded image on a HAM-E since the IFF loader does not disturb the embedded control

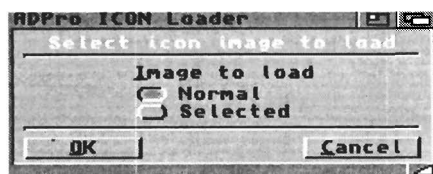


Figure 6.10: The ICON loader control panel.

information in the file. The HAME loader actually strips away the embedded control information after using it to convert the encoded image into raw 24 bit-plane color or 8 bit-plane gray data.

6.19 ICO

NOTE The ICO loader was developed by Steve Tibbett, and not ASDG, Incorporated. As such, you should contact Steve Tibbett's BBS at (613) 731-3419 for technical support. This module is included in the ADPro package under agreement with Steve Tibbett.

The ICO loader allows you to import Microsoft WindowsTM icon (.ico) files into ADPro. These images are always 766 bytes long with dimension of 32 x 32 pixels.

Upon launching this loader, a file requester will appear from which the ICO file may be chosen.

6.20 ICON

NOTE The ICON loader was developed by Steve Tibbett, and not ASDG, Incorporated. As such, you should contact Steve Tibbett's BBS at (613) 731-3419 for technical support. This module is included in the ADPro package under agreement with Steve Tibbett.

The ICON loader allows you to import the primary image in an Amiga icon (.info) file.

Upon launching this loader, a file requester will appear from which you can choose the ICON file to load. Be sure that the **Pattern** field in the file requester is not filtering out ".info" files, otherwise you won't see the icon files! Note that all icon files must contain a ".info" extension to be used with this loader.

If the icon contains a selected or alternate image (a separate image that is displayed when the icon is clicked on), then a control panel (as shown in Figure 6.10) will appear. Here, you can choose which image (**Normal** or **Selected**) to load. If no alternate image exists, then the icon imagery will load immediately

Select the **OK** button to load the icon, or **Cancel** to exit the loader without loading an image.

6.21 IFF

The IFF loader (also referred to as the Super-IFF loader) allows you to load any properly defined IFF-ILBM picture file, including all of the following:

- Commodore standard IFF files in 1 through 5 bit-planes, corresponding to 2 through 32 colors, respectively.
- Commodore standard IFF files in the 64 color Amiga Extra-Halfbrite (EHB) format.
- Commodore standard IFF files in the 4096 color Hold-And-Modify (HAM) format.
- IFF files in Sliced-HAM (SHAM) format (can be read but not written).
- IFF files in A-HAM or Dynamic HAM formats (4096 colors).
- IFF files in A-RES or Dynamic Hi-Res formats (4096 colors in hi-res).
- Commodore standard IFF files in 12, 15, 18, 21, and 24 bit-planes (4096, 32768, 262144, 2097152, and 16777216 colors, respectively). This includes 21 bit-plane files created by Digi-View 4.0.
- DCTV images.

The Super-IFF loader will look for a palette in the image file. If there is one, it will be tested to see if it contains only grayscales. If it does, the image will be loaded into 8 bit-planes and will be treated as grayscale image data. Otherwise, the image will be loaded into 24 bit-planes and treated as a color image.

NOTE When attempting to load an image already in a rendered format (such as the 1 to 6 bit-plane formats) and there isn't enough memory to convert the full image into 24 bit-planes or 8 bit-planes if grayscale), the Super-IFF loader will simply load the image and not convert it into a deep bit-plane format. This means you will be able to view the image, edit its palette, and even save the image back out in a file format supporting the same depth, but you will not be able to perform any other of ADPro's image processing functions.

NOTE IFF images wider than 4096 pixels (for images greater than or equal to 12 bit-planes) or 5440 pixels (for images less than 12 bit-planes), or taller than 5440 pixels (any depth) cannot be loaded in.

6.22 IMPULSE

The IMPULSE formats are defined by Impulse, Inc. for use with their color products, such as Turbo-Silver and Imagine. ADPro's IMPULSE loader supports reading the RGBN and RGB8 IMPULSE formats. The RGBN format stores 4 bits per primary color, giving a total of 12 bit-planes. The RGB8 format stores 8 bits per primary, giving a total of 24 bit-planes of color information.

To read an image in either format, select the IMPULSE loader and choose the file from the file requester.

Files in the IMPULSE format are always interpreted as color images.

6.23 IV24

The IV24 (Impact Vision 24) from Great Valley Products provides a 24 bit-plane framegrabber as well as being a 24 bit-plane display device. The IV24 loader provides control over the IV24's framegrabbing capability.

NOTE You must have an IV24 installed in your machine and you must have installed the **fye.library** in your **LIBS:** directory to make use of this loader. If these conditions are not met, selecting this loader will cause an error message to be displayed on ADPro's main window.

The control panel shown in Figure 6.11 is presented when you execute the IV24 loader. You can use the **Width:**, **Height:**, **X Offset:**, and **Y Offset:** controls to define a rectangle within which the image data will be grabbed.

The upper left hand corner of the IV24 is at an **X Offset:** of 0 and **Y Offset:** of 0.

Note that the **Width:** plus the **X Offset:** must be less than or equal to 768. The **Height:** plus the **Y Offset:** must be less than or equal to 566.

If your video set-up allows you to view the IV24 display on your Amiga monitor, selecting the **View** button will switch to IV24 video. Clicking the left mouse button while watching IV24 video brings you back to Amiga video.

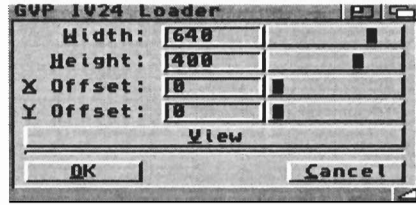


Figure 6.11: The IV24 loader control panel.

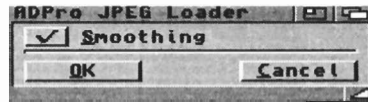


Figure 6.12: The JPEG loader control panel.

Note that while watching IV24 video (as a result of pushing the button marked **View**) you can press the **Spacebar** key on your keyboard to cause a framegrab.

Selecting the **OK** button causes a framegrab to take place. Selecting the **Cancel** button exits the IV24 loader with no action taking place.

NOTE This loader does not support composition nor landscape style loading.

6.24 JPEG

The JPEG loader can read and decompress images written to the JFIF file format. The JFIF format is a standard for storing images compressed using the Joint Photographic Expert Group compression algorithms. JPEG can compress images to a fraction of their original size but does so with some loss to the image. That is, the decompressed image may vary slightly from the original image.

The loader can also read HSI style JPEG files. These files have a mixture of GIF and JPEG encoding styles.

The JPEG loader control panel is shown in Figure 6.12. The **Smoothing** (checkbox) controls whether or not smoothing is enabled. JPEG actually compresses small square regions of the image individually. Sometimes, when an image is very heavily compressed, you may see evidence of these small squares after decompression. Smoothing is a blurring operation which takes place during decompression and, therefore, is written so that it does not blur the entire

image, but only the borders between the small squares. This can often mask or hide some of the artifacts which JPEG compression can introduce.

Pressing the **OK** button will cause the file requester to appear. You can use it to select the file to be read.

6.25 MACPAINT

The MACPAINT loader interprets files in the MacPaint format. This format provides for black and white (1 bit-plane) images only. Files are converted to 8 bit-plane gray scale as they are loaded.

6.26 PAR_PEG

NOTE The PAR_PEG loader was developed by Digital Processing Systems (DPS), and not ASDG, Incorporated. As such, you should contact DPS for technical support on this loader. This module is included in the ADPro package under agreement with DPS.

You will need version 1.5 or later of the PAR software to use this loader.

Digital Processing Systems' PAR_PEG is a loader module for ADPro, which can import still and frames from animation files on a Personal Animation Recorder (PAR) hard drive.

NOTE The PAR software must be running at the same time for this loader to work.

Upon launching the loader, a file requester will appear from which you can select a still frame or animation from the PAR hard drive (assigned as **DDR:**). Once selected, the PAR_PEG control panel (as shown in Figure 6.13) will appear.

From this control panel you can choose which frame to load and how to load it. If you selected a still frame, then only the **Mode:** cycle gadget will be available. Otherwise, the **Frame:** selector will be available to specify the frame in the animation to load.

The **Mode:** cycle gadget can be set to one of three choices. If set to **Frame**, the entire frame (752 x 480) will be loaded. Setting it to **Field 1** or **Field 2** will load that particular field (752 x 240) of the frame.

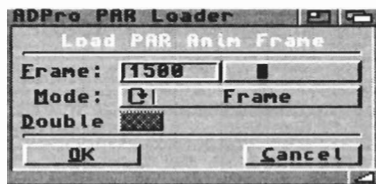


Figure 6.13: The PAR_PEG loader control panel. For still frames, the **Frame:** control is not available.

When loading a field, the **Double** checkbox will be enabled. Checking this box will double each scanline in the field so that the image fills one “frame” area. If you just want an unmodified field, uncheck this box.

Select the **OK** button to load the image, or **Cancel** to exit the loader without loading anything.

6.27 PCX

The PCX format was defined by the ZSoft Corporation for use in their product, PC Paint Brush. PCX images may contain from 1 to 256 colors, including special palettes for IBM PC EGA and CGA display boards. PCX images may now also contain 24 bit-planes of color information.

The ADPro PCX loader will correctly interpret most PCX images including those with CGA and EGA palettes. However, some CGA images may contain misleading palette information which ADPro cannot help but follow.

To read an image in the PCX format, select the PCX loader and choose the PCX file from the file requester.

6.28 POINTER

The POINTER loader can be used to digitize any mouse pointer sprite which can be displayed while this loader is running. Those applications which do not multitask (such as some games) cannot have their mouse pointer's digitized since ADPro cannot run at the same time. Being able to digitize the mouse pointer adds realism to screen grabs performed with the SCREEN loader.

Selecting the POINTER loader displays its control panel (shown in Figure 6.14). At this time, a hot-key sequence is installed in the input stream of your Amiga. When you press the **Right-Alt + Right-Shift + Backspace**

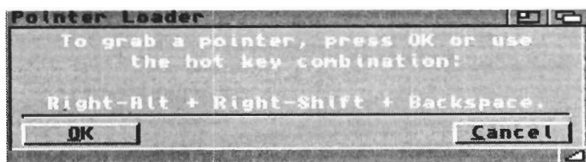


Figure 6.14: The POINTER loader control panel.

key sequence, the mouse pointer currently being displayed will be digitized.

If compositing mode is enabled, you will have an opportunity to place the newly digitized pointer over a predefined image already in ADPro's memory. Note that pointers are always digitized in low resolution (or high resolution, if running under Workbench 3.0 or later) pixels and ADPro has no way of knowing what size pixel you intend it to maintain within its memory. Therefore, some combinations of original screen resolutions can lead to a slightly distorted composition of the mouse pointer.

When compositing with the mouse pointer, its transparent color is automatically discarded to allow the digitized mouse pointer to correctly overlay the background image.

NOTE To properly composite (transparent color is actually transparent) the imagery of a pointer, you should set the R, G, and B transparency values (in the Compositing control panel) to 1, 1, 1. Anything that is transparent in the pointer imagery (even colors that are 1, 1, 1) will be transparent during compositing.

6.29 QRT

The QRT format is used by several exceedingly powerful ray tracing programs such as DKB Ray Trace and the QRT ray tracer. The format itself is quite simple, actually being a single file cousin to the SCULPT format.

QRT images are 24 bit-plane color only. A drawback of the QRT format is that image data written in this format is not compressed in any way. Therefore, QRT files can be quite large.

To read an image in the QRT format, select the QRT loader and choose the QRT file from the file requester.

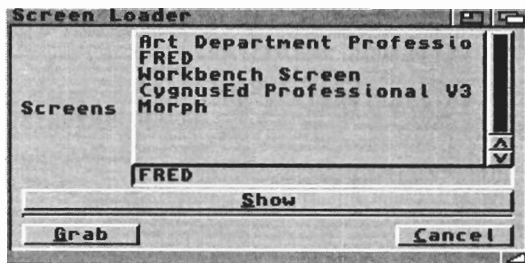


Figure 6.15: The SCREEN loader control panel.

6.30 SCREEN

The SCREEN loader is an example of the versatility of the ADPro loader/saver interface. It doesn't load an image from disk as the other standard loaders do. Rather, it captures the rendered image data from another Amiga screen and converts this data for use by ADPro.

Selecting the SCREEN loader causes its control panel to appear. In this panel is a vertically scrollable list of the currently defined Amiga screens. Each screen will be listed by its Intuition screen name. If a screen does not have an Intuition screen name, then its name will be presented as **No Name**. If more than one screen has no Intuition name, each **No Name** will be differentiated by a sequence number.

Select a screen to capture by clicking on the button containing the screen's name. You may confirm that this is the screen you intend to capture by clicking on the **Show** button. The selected screen will be brought to front. A click of the left mouse button will bring ADPro back in front.

There are two ways to actually load the selected screen. First, you may select the **Grab** button after selecting one of the screen choices in the SCREEN loader control panel. The selected screen will be loaded into ADPro and can then be processed as if it were any other type of loaded image.

The first method of screen grabbing cannot allow you to grab a screen while that screen is showing menus. This is because you (the user) can't be in two places at one time (i.e., pushing the grab button and holding down the menu button in the screen you wish to grab).

The second method allows you to grab a screen including its menus.

When the SCREEN loader begins executing, it temporarily installs the hot key sequence, **Right-Alt + Right-Shift + Backspace**. When the SCREEN loader exits, it removes the hot key sequence from the system's input chain.

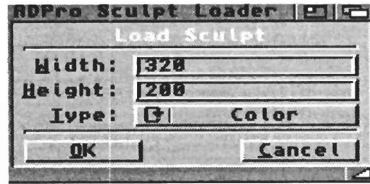


Figure 6.16: The SCULPT loader control panel.

You can grab any screen including its menus by selecting the SCREEN loader and then bringing the desired screen to front, causing the desired menu to appear, and then depressing the hot key sequence. When you release the keys (while still holding down the right mouse button), the active screen will be loaded, including the active menus.

6.31 SCULPT

The SCULPT format was actually defined by Mimetics Corporation for use with their 24 bit-plane frame buffer. The format is better known as the one produced by Byte-By-Byte's SCULPT series of 3D modelling products.

The SCULPT format has no header information at all contained in the image itself. Therefore, in the SCULPT format, *you* must remember the exact dimensions of an image in order for it to correctly load.

After selecting the SCULPT loader, its control panel (as shown in Figure 6.16) will appear. You will be asked to enter a width, height, and type of the image to be loaded. Remember, that the SCULPT format does not store the width and height as header information in the data files. Therefore, you must keep track of these values yourself.

The SCULPT loader can load a combination of three files to form a color image, or load a single file to create a grayscale image. After specifying a width and height in the **Width:** and **Height:** fields, select which type of image (color or grayscale) you wish to load.

If loading a color image, you will be asked to specify three image files containing the red, green, and blue information for the image respectively.

ADPro will check the size of each of the three files you specified against the width and height you provided. If the size of one of the specified files does not match the size indicated by the entered width and height, the load will be aborted.

There is a naming convention for images stored in the SCULPT format. This

convention dictates that each of the three raw image files comprising one SCULPT image share the same root file name and have the **red**, **grn**, and **blu** extensions for the red, green, and blue components, respectively.

When you specify the name of the red component, if you do not specify a filename extension or if the extension you specify is **red**, then the SCULPT loader will automatically generate the conventional extensions for the green and blue components.

The Mimetics frame buffer software also uses **ored**, **ogrn**, and **oblu** extensions to signify that the image in question is overscanned. The SCULPT loader will automatically generate these extensions where appropriate as well.

When loading a grayscale file, you will be asked to select only one file in the SCULPT format.

6.32 TEMP

The TEMP loader allows you to load the image currently in the ADPro temporary (undo) buffer into the primary image buffer. If no previous image was saved to the temporary buffer, then the image currently in ADPro will remain unchanged. For a discussion of saving to this temporary buffer, please read Section 7.29.

Besides using this buffer for “undo” operations, you can also use it for quick retrieval of the same image. One application of this technique would be to create various versions of the same image. You would save the image with the TEMP saver, make the first version’s changes, and save it out with one of the savers. Select the TEMP loader (which loads the original image from the temporary buffer), make the second version’s changes, and save those out. Continue selecting the TEMP loader for subsequent versions.

Also, the TEMP loader retrieves the pixel aspect of the image in the temporary buffer.

There are two ways of determining if image data is currently in the temporary buffer:

1. by selecting **About...** from the **Project** menu to display the information about the program and seeing if an asterisk (*) appears to the right of the **Buffer Size:** value.
2. by scrolling the **Image Information** list text and seeing what percentage of the allocated image buffer is occupied by “temp” data.

6.33 VLAB

NOTE The VLAB loader was developed by MacroSystem, and not ASDG, Incorporated. As such, you should contact MacroSystem for technical support. This module is included in the ADPro package under agreement with MacroSystem.

The VLAB loader works generally the same way as the corresponding commands in the VLab software. Consult your VLab documentation for more information. The VLAB loader is just another way of getting the same functionality from the VLab system, but from the ADPro interface. This removes the step of saving the digitized image to disk, then loading it into ADPro.

6.34 YUVN

NOTE The YUVN loader was developed by MacroSystem, and not ASDG, Incorporated. As such, you should contact MacroSystem for technical support. This module is included in the ADPro package under agreement with MacroSystem.

Also, you must have the `vlab.library` installed in your `LIBS:` directory. This file is installed if you select this loader during the installation process.

The YUVN file format stores YUV pictures which are mainly used in broadcast television. YUVN supports CCIR-601-2, which is a worldwide standard for NTSC and PAL television.

As with many other loaders that import images, a file requester will appear asking for the name of the YUVN file to load. Once selected, a meter window will appear atop the ADPro window showing the progress of the load operation. Once completed, you may manipulate or save the image with any of the other ADPro controls.

6.35 LoadNShowFC24

This pseudo loader allows you to select and automatically view an image on an Impulse FireCracker24 graphics board.

NOTE Do NOT try to execute this pseudo loader if you don't have a FireCracker24 graphics board installed in your machine.

6.36 `_LoadNShowOV`

This pseudo loader allows you to select and automatically view an image on a Centaur Development / Opal Technology OpalVision graphics board.

NOTE Do NOT try to execute this pseudo loader if you don't have an OpalVision graphics board installed in your machine.

6.37 `_LoadNShowPICASSO`

This pseudo loader allows you to select and automatically view an image on a Village Tronic Picasso II graphics board.

NOTE Do NOT try to execute this pseudo loader if you don't have a Picasso II graphics board installed in your machine.

6.38 `_LoadNShowRETINA`

This pseudo loader allows you to select and automatically view an image on a MacroSystem GmbH Retina or Retina Z-III graphics board.

NOTE Do NOT try to execute this pseudo loader if you don't have a Retina/Retina Z-III graphics board installed in your machine.

6.39 `_VT_Grab`

This pseudo loader allows you to grab the contents of either the DV1 or DV2 framebuffer of the Video Toaster.

NOTE Do NOT try to execute this pseudo loader if you don't have a Video Toaster and its software installed in your machine.

Chapter 7

Savers

This chapter provides detailed descriptions of the use and operation of all savers which are included with ADPro. ARexx control of these savers is described in Chapter 12.

7.1 A2410

The A2410 saver gives you the ability to directly control the A2410 display board from Commodore. This board provides a high quality 8 bit-plane RGB video framebuffer.

NOTE You must run Commodore's TIGA initialization software before the A2410 can be used. See your A2410 documentation for more details.

The A2410 display board requires the presence of rendered image data containing from 1 to 8 bit-planes in a standard palette mapped configuration. That is, the rendered image data cannot be in one of the rendering modes unique to the Amiga such as HAM (Hold-And-Modify) or EHB (Extra-Halfbrite).

This saver is slightly different from some of the other savers compatible with ADPro in that it allows direct and continuous control over a hardware device, allowing multiple operations to be performed without exiting the saver. With a saver such as the one for GIF, the saver is invoked and a file gets saved. With the A2410 saver, the same image (or different parts of the same image) can be saved to the A2410 multiple times without leaving the saver.

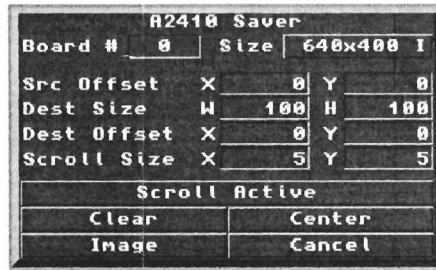


Figure 7.1: The A2410 saver control panel.

Screen Resolution	Interlace	Crystal Required (MHz)
640 x 400	Yes	14.318
736 x 480	No	26.6363
800 x 600	No	36.00
1024 x 768	Yes	44.98
1024 x 1024	No	67.88
1024 x 768	No	67.88
1024 x 1024	No	89.2108

Table 7.1: Resolutions supported by the A2410 and the crystals required to implement them.

The A2410 saver control panel is shown in Figure 7.1. Along the top line, the button marked **Board #** allows you to toggle through which of possibly many A2410's you wish to control right now. If you have only a single A2410 in your system, this button will display a value of 0 only.

The A2410 contains two on-board crystals defining which two of several preset resolutions will be available. The button marked **Size** allows you to toggle between the two screen configurations that an A2410 can support at any one time. Table 7.1 lists the different standard screen resolutions possible with the A2410 along with the crystals required to implement those resolutions.

The **Center** button automatically adjusts all of the sizing and offset controls of the A2410 saver so that the center of the image data will coincide with the center of the A2410, taking into account the currently selected screen resolution.

The button marked **Clear** will cause the entire screen to be cleared to black.

Selecting the button marked **Image** causes rendered image data to be transferred to the A2410. The amount of data transferred as well as where the

image will be placed is controlled by the offset and size settings.

The **X** and **Y** values of **Src Offset** allow you to select the upper left hand corner in the source image data (the image in ADPro memory) from which to start transferring image data to the A2410. In general, when an image is smaller than the currently screen resolution, these values will be set to zero as they are in Figure 7.1. Setting these values to non-zero has the effect of “cropping” the image actually sent to the A2410 without actually disturbing the image data in ADPro memory.

The **Dest Size** values determine how large an area on the A2410 will be written into. This also determines how large an area from the source image (the image in ADPro memory) will be accessed. In the example of Figure 7.1, selecting the **Image** button will transfer a 100 by 100 pixel area from ADPro memory into a 100 by 100 pixel area in A2410 memory.

The **Dest Offset** values determine the upper left hand corner of the area which will be written in A2410 memory. This allows you to place an image into an arbitrary place in the A2410 display. For example, to center (by hand rather than using the **Center** button) a 320 by 200 pixel image on the A2410 display (when set to 736 by 480) you would set the **Dest Offset X** to 208 and the **Dest Offset Y** to 140.¹

When scrolling is active, you can set the size of the basic scroll by placing values in the **Scroll Size** settings. These values determine how many pixels to the left or right (the **X** setting) or up and down (the **Y** setting) the image will move when the cursor keys are struck.

The **Scroll Active** button is necessary to tell the saver that you intend to use the cursor keys for scrolling when this control panel is displayed. Otherwise, the cursor keys will be used to move the cursor in the currently active input field. To use the cursor keys to scroll, select the button marked **Scroll Active** so that it remains highlighted. Now, depressing the cursor keys will cause the image on the A2410 to scroll in the specified direction. The area that will scroll is defined by **Dest Size**.

Note that the scrolling takes place only within the window you specified using the **Dest Size** and **Offset** fields.

If you depress one of the cursor keys while the **Shift** key is also depressed, you will scroll three times the distance specified in the **Scroll Size** input fields. If you hold down the **Ctrl** key at the same time as one of the cursor keys, you will scroll to the far edge in the direction you specified.

¹This is $(736 - 320)/2$ for the x offset and $(480 - 200)/2$ for the y offset.

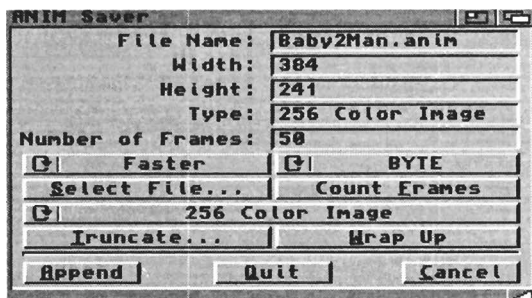


Figure 7.2: The ANIM Saver control panel. The **Count Frames** button has already been selected (**Number of Frames:** is not zero), causing it to be ghosted.

7.2 ANIM

The ANIM saver allows you to save the current rendered image data (either the cropped screen or entire size) as the last frame of an IFF-ANIM file or as the first frame of a newly-created ANIM. Raw data (8 and 24 bit-plane IFF) is not currently supported in the IFF-ANIM file specifications.

NOTE This version of the ANIM saver supports the new ANIM OpCode-8 (ANIM8) format, which allows for word and long word compression, in addition to the byte compression method used in ANIM5).

After selecting the ANIM saver, its control panel will appear, as shown in Figure 7.2.

The first step you need to do is select the IFF-ANIM file onto which the current image data will be appended. Select the **Select File...** button. A file requester will appear to allow you to select the ANIM file. If you select a file that is a valid IFF-ANIM, the current rendered image data will be appended to it. If you enter the name of a non-existent file, the ANIM saver will create a new IFF-ANIM file, using the rendered data as the first frame of the animation.

The upper half of the control panel is where information about the selected ANIM file will be displayed. To display the number of frames in the animation, select the **Count Frames** button. Note that this process may take some time to complete, especially for ANIM files that contain a lot of frames. Once you select an ANIM file, its filename will appear to the right of the **File Name:** label and the width, height, and type of each frame will be displayed next to the **Width:**, **Height:**, and **Type:** labels. After retrieving the frame count, this button will become unselectable (the button's image will be ghosted).

As previously stated, ANIMs can only be made up of rendered image data. The type of rendered data can be selected from the **Save Type** cycle gadget. Clicking on this button will toggle between **Cropped Screen Image** and **n Color Image**, where n is the number of colors in the rendered data.

The Compression Quality cycle gadget allows you to control the speed which the ANIM will be encoded and saved to disk. If set to **Smaller**, the saver will optimize the ANIM for file size. If set to **Faster**, the encoding of the ANIM will be faster; this is the “no size optimization” setting. The difference in playback speed between these two settings is negligible.

The Compression Type cycle gadget controls the method used to compress the image data. **BYTE** gives you a traditional OpCode 5 ANIM file, whereas **WORD** and **LONG** give you OpCode 8 files. Although byte compression will probably result in smaller files, word and long word compressed files will play faster on any Amiga machines, but will play up to twice as fast on AGA machines, such as the A4000 (assuming you have an ANIM8-compatible player).

For short-frame ANIMs, the values you use for Compression Quality and Type might not have a substantial effect on the size and playback speed. Noticeable differences will only be apparent for multi-frame ANIMs played on these new Amiga modules with 32-bit chip (graphics) memory access.

If the current rendered image data does not match all of the selected ANIM's attributes (width, height, and type), a message will appear. The rendered data must be the same as what the ANIM expects, otherwise this saver will not be able to append it to the file.

Not only does the ANIM saver allow you to append an image to an ANIM file, it allows you to modify an existing ANIM file on disk.

The **Truncate...** button allows you to lessen the ANIM to fewer frames than it currently has. Selecting this button will bring up another panel with a integer input field. You can either enter a positive or negative number. If the number is positive, the ANIM will be truncated to that many frames. If you enter a negative number, that many frames will be removed.

NOTE You will need to perform a **Count Frames** operation before a negative number can be accepted for the **Truncate...** value. From ARexx, you do not need to do this step.

The **Wrap Up** button will copy the first and second frames of the animation to the end of the file. This is part of the IFF-ANIM file format specification and is used to produce a smooth looping animation. This operation will close the ANIM file. This means that the next time you enter this saver, you will have to select a new ANIM file to work on.

To actually append the current rendered image data, select the **Append** button. The meter window will appear, showing the progress of the save. If any errors occur, a message will be displayed. If no errors occurred, the control panel will disappear.

A time-saving feature of the ANIM saver is that it remembers the position of the last frame in the ANIM file. This allows you to append successive frames without having to wait for the saver to find the end of a file for every new frame. For animations made up of a few frames, this feature might not be as evident as with a “multi-frame” animation.

To exit the saver without actually appending the image, select **Cancel**. The control panel will disappear, returning control back to the main window. If you want to use the time-saving feature just described, select this button to exit the control panel.

If you want to exit the control panel without saving the information about the last frame's position, select the **Quit** button. **Quit** will close the ANIM file, allowing you to use it in other programs.

Note that the ANIM saver will store the current screen mode with the saved frame.

NOTE If you want to play the animation with an ANIM viewer, you **MUST** select the **Quit** button after finishing all of your modifications with ADPro, select a different animation file using the **Select File...** button, select **Wrap Up**, or exit ADPro altogether. If you fail to do any of the steps, other programs will not be able to use the file.

7.3 BMP

The BMP format is the standard image file format used by Microsoft Windows. BMP files may come in one of four depths: 1, 4, 8 or 24 bit-planes. The BMP saver supports saving, raw, rendered, and screen image data. You will be asked to select one of the following:

- **Cropped Screen Image** — Selecting this mode will save the screen image data which would be displayed if you select the **Redisplay** button. This choice is available only if you have rendered in a non-Amiga specific displayable mode.
- ***n* Color Image** — Selecting this mode will save the entire rendered image. The number of colors in the rendered image will be presented in place of the *n*. This choice is available whenever rendered image data is available in a non-Amiga specific format.

- ***n* Bit-Plane Raw Image** — Selecting this mode will save the entire raw image as either a 24 bit-plane color image or 8 bit-plane grayscale image.

Note that the BMP saver saves only in the uncompressed BMP format (although the BMP loader will read both compressed and uncompressed BMP files). This is because we have found that most Windows applications cannot read compressed BMP files. Also, saving 24 bit-plane BMP files using their compression will almost certainly produce a larger file than the original.

When the file requester appears, use it to select the name of the file under which a BMP file will be saved.

Note that you will benefit from use of the enhanced palette mode when saving rendered images in the BMP format.

7.4 CDXL

The CDXL saver allows you to create, modify and extend CDXL files. It allows you to specify the amount of space to leave for audio and padding. It, in conjunction with the CDXL loader, can even be used to copy audio and pad data from other CDXL files.

NOTE This file format is used by CDTV/CD32 developers in the application making process. The use to non-developers will probably be limited to an alternative file format, as there is no distributable CDXL viewer included with this package.

The saver may be used with 1 to 8 bit rendered data, EHB, HAM, and HAM8. It also supports 8 bit gray and 24 bit color chunky image data.

Upon invoking the saver, a file requester will appear for the specification of a filename. You can append to the end of an existing CDXL file, replace a particular frame, or start a new file.

If the specified filename does not exist on disk, the initial control panel (shown in Figure 7.3) will appear. It is from this panel that you set up characteristics of the CDXL file.

CDXL files can contain both video and audio. The audio and padding controls are only appropriate for CDTV/CD32 developers. Developers should consult their documentation for more information on what these controls do.

If you are currently working with a CDXL file, an alternate control panel (shown in Figure 7.4) will appear. With fewer controls, this is where you specify which frame to save to.

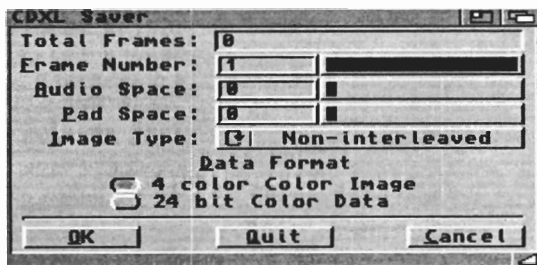


Figure 7.3: The CDXL Saver control panel for new files.

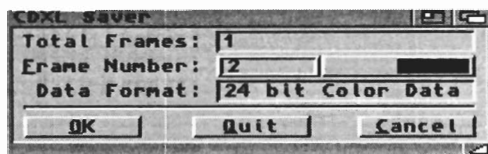


Figure 7.4: The CDXL Saver control panel for existing files.

The **Total Frames:** field shows the number of frames in the CDXL file. You can specify which frame to save to with the **Frame Number:** input field. If you are creating a new file, then this value cannot be changed (it will stay at 1). You are allowed to enter a frame number one larger than the number of frames in the file, thus extending the file by one frame.

Below the audio and pad controls is the **Image Type:** cycle gadget (only available if you are creating a new CDXL file). This control is only appropriate for CDTV/CD32 developers. For non-developers, these controls can be left at their default settings.

Beneath this control are the radio buttons which allow you to pick rendered or raw data.

A time-saving feature of the saver is that it stays resident in memory, remembering the position of the last frame in the CDXL file. This allows you to append successive frames without having to wait for the saver to find the end of a file for every new frame, thereby processing animations extremely quickly. For animations made up of a few frames, this feature might not be as evident as with a "multi-frame" animation.

To save the image to the specified frame, select the **OK** button.

To exit the saver without actually replacing or appending the image but keeping the current CDXL file open and the module resident in memory, select **Cancel**.

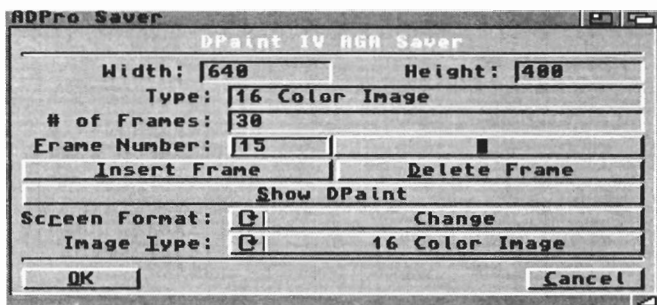


Figure 7.5: The DPaint IV AGA saver control panel.

If you want to exit the control panel without saving the information about the last frame's position, select the **Quit** button. **Quit** will close the CDXL file, allowing it to be accessed with other CDXL-supported applications.

7.5 CLIPBOARD

The CLIPBOARD saver allows ADPro to write pictures from the Amiga's Clipboard device. The Clipboard is a resource which can be shared between applications which support it. Note that text as well as images can be placed into the clipboard. The CLIPBOARD saver will write only picture type information.

Note that the Amiga's Clipboard device is not widely supported, and those applications that do support the Clipboard may not be expecting any other type of image data to be in the Clipboard except standard Amiga displayable images. Also note that the Clipboard device is often assigned to **RAM:**. This means that any image data written to the Clipboard will consume memory.

7.6 DPAINT

The DPAINT saver (shown in Figure 7.5) allows you to save the current rendered image in ADPro's image buffer to the primary image, swap buffer image, or ANIM frame buffer to the currently running copy of Electronic Arts' Deluxe Paint IV AGA paint program.

NOTE While this saver's panel is displayed, you **cannot** perform any function within DPaint.

The **Width:** and **Height:** text fields display DPaint's current page dimensions.

The **Type:** text field displays the type of image that DPaint can accept.

The **# of Frames:** text field displays two more than the total number of ANIM frames. One of the extra frames (frame 0) is for DPaint's swap buffer, while the other frame (frame n) is for a possibly new frame at the end of the ANIM. This ability is given so that you do not need to manually add a frame (using the **Insert Frame** button), move the **Frame Number:** slider to the rightmost position, then press the **OK** button.

The **Frame Number:** slider allows you to specify the swap buffer (0) or a specific ANIM frame (1 through n) into which the current image in ADPro's image buffer will be saved. The maximum frame number is actually one more than the total number of DPaint ANIM frames. This allows you to append to the end of the ANIM without having to first insert a frame (described below).

The **Insert Frame** button adds a new, blank frame immediately after the current frame in the DPaint ANIM sequence, whereas **Delete Frame** removes the current frame (defined by the **Frame Number:** slider value) from the ANIM. With **Insert Frame**, the inserted frame is a copy of the current frame at the time the button was selected.

NOTE You cannot delete frame 0, the swap buffer.

The **Show DPaint** button displays the current frame in DPaint, giving you a preview of what may be overwritten if the save operation is performed. You can press the left mouse button to return to the DPaint saver control panel.

NOTE From within the DPAINT saver you *can* differentiate between the contents of the swap page and the normal page just by setting the **Frame Number:** slider and clicking on the **Show DPaint** button, but from DPaint itself—if you do not have an ANIM loaded—then you *will not* know the difference.

The **Screen Format:** cycle gadget specifies if DPaint's screen format should be modified to match the image to be saved. This cycle gadget will be ghosted (unselectable) in the **Change** setting if ADPro differs from DPaint in the following ways:

- Different pixel depth.
- A mismatch of HAM versus EHB versus normal modes.
- DPaint has an ANIM loaded and it is not the same size as the image to be saved.

In these cases, DPaint has to change its screen format to accept the image in ADPro.

If unghosted, then you can select between **Change** and **Preserve**. If set to **Change**, every effort will be made to switch DPaint into the same screen format as ADPro. If set to **Preserve**, then DPaint will not change screen format. In most cases and especially if you are building an ANIM, you should use **Preserve**.

The **Image Type:** cycle gadget allows you to select the type of image data available in ADPro to send to DPaint. You can select from the entire rendered image data or just the current “screen-full” (the portion of the image which is viewable if the **Redisplay** button on ADPro’s screen is selected).

Press **OK** to perform the save, or **Cancel** to abort the save operation. If you are saving to a frame that is not a “new” one—you are not appending a frame—then the frame count will stay the same as before the save operation.

Note that if you are trying to save (**OK**) a new ANIM frame but DPaint’s page size is larger than the ANIM frame size, you will get a requester saying that a page cannot be inserted. This will be the same type of error you would get in DPaint if you tried to do the same operation.

7.7 DPIIE

You must have rendered data available in order to save in the DPIIE format. To use the DPIIE format, the image must have been rendered with 256 colors. If you wish to save an image for use with Deluxe Paint II Enhanced with fewer colors than 256, you can use the standard IFF saver.

After selecting the DPIIE saver, use the file requester to specify the filename to give the saved rendered image.

Note that you can benefit from use of the enhanced palette mode when saving images for use with Deluxe Paint II Enhanced.

7.8 FC24

The FC24 saver gives you the ability to directly control the FireCracker 24 display board from Impulse. This board provides a high quality 24 bit-plane RGB video framebuffer. Any work-in-progress within ADPro (both raw and rendered data) can be saved to the FireCracker 24.

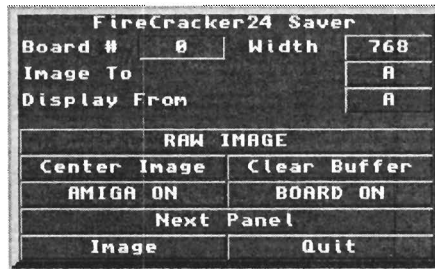


Figure 7.6: The first of two control panels for the FireCracker 24 saver.

NOTE This saver module has more control than the FC24 display module, but requires that you switch your saver module. If you want a quick method of displaying an image to the FireCracker 24, you might want to use the display module support instead. See Section 9.3 for more information on using this display module. This saver module is also included for compatibility with ARexx scripts which require the presence of a FC24 saver module.

This saver is slightly different from some of the other savers compatible with ADPro in that it allows direct and continuous control over a hardware device allowing multiple operations to be performed without exiting the saver. With a saver such as the one for GIF, the saver is invoked and a file gets saved. With the FC24 saver, the same image (or different parts of the same image) can be saved to the FireCracker 24 multiple times without leaving the saver.

The FC24 saver presents two control panels. The first is shown in Figure 7.6. Along the top line, the button marked **Board #** allows you to toggle through which of possibly many FireCracker 24's you wish to control right now. If you have only a single FireCracker 24 in your system, this button will display a value of 0 only.

The button marked **Width** allows you to toggle through the four board widths which the FireCracker 24 supports. These include 384, 512, 768, and 1024 pixels across.

The FireCracker 24 supports double buffering when set to either of its two lower resolutions (384 or 512 pixels wide). In its higher resolution modes (768 or 1024 pixels across) the FireCracker 24 does not support double buffering.

When double buffering is supported (i.e., the **Width** button set to 384 or 512), you can select which buffer will be written to by the next **Image** command by toggling the button marked **Image To**. This button toggles between an **A** and **B** setting corresponding to the two addressable buffers.

Similarly, you can choose which buffer is to be displayed by clicking on the button marked **Display From**. Like the button marked **Image To**, this button is enabled only when the FireCracker 24 width is either 384 or 512.

The button marked **RAW IMAGE** can be toggled between that setting and **RENDERED IMAGE**, provided that each type of image data is available. Using this button (and assuming both image data types are available) you can select which type of image data will be saved to the FireCracker 24.

The **Center Image** button automatically adjusts all of the sizing and offset controls of the FireCracker 24 saver so that the center of the image data will coincide with the center of the FireCracker 24, taking into account the currently selected width.

The button marked **Clear Buffer** will cause the entire buffer currently selected to be cleared to black.

The FireCracker 24 allows Amiga video to be looped through and combined with FireCracker 24 video for display on a single monitor. If you take advantage of this feature (to use a single monitor for both video outputs) then there are times at which you might want to disable either of the Amiga video or the FireCracker 24 video. The buttons marked **AMIGA ON** and **BOARD ON** can be toggled to enable or disable the respective video source.

Note that when the Amiga's video is disabled, all input (via mouse and keyboard, except for the cursor keys) will be disabled until either the left mouse button or the space bar is pressed. Note also that disabling the FireCracker 24 video while the Amiga video is also disabled will automatically reenable the Amiga video.

Selecting the button marked **Next Panel** takes you to the second FireCracker 24 saver control panel which is described below.

Selecting the button marked **Image** causes image data to be transferred to the FireCracker 24. The type of data is determined by the setting of the button marked **RAW IMAGE** in Figure 7.6. The amount of data transferred as well as where the image will be placed is controlled by settings found on the second FireCracker 24 control panel.

The second control panel for the FireCracker 24 saver is shown in Figure 7.7. It has several controls in common with the other FireCracker 24 control panel and such controls perform the same functions. They are repeated on both panels for your convenience.

The **X** and **Y** values of **Src Offset** allow you to select the upper left hand corner in the source image data (the image in ADPro memory) from which to start transferring image data to the FireCracker 24. In general, when an image is smaller than the currently selected FireCracker 24 width, these values will be set to zero as they are in Figure 7.7. Setting these values to non-zero has

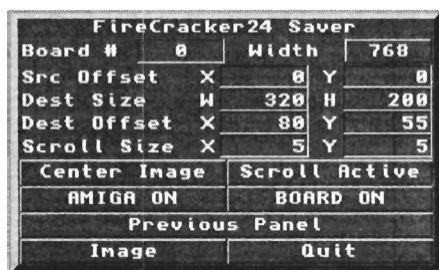


Figure 7.7: The second of two control panels for the FireCracker 24 saver.

the effect of “cropping” the image actually sent to the FireCracker 24 without actually disturbing the image data in ADPro memory.

The **Dest Size** values determine how large an area on the FireCracker 24 will be written into. This also determines how large an area from the source image (the image in ADPro memory) will be accessed. In the example of Figure 7.7, selecting the **Image** button will transfer a 320 by 200 pixel area from ADPro memory into a 320 by 200 pixel area in FireCracker 24 memory.

The **Dest Offset** values determine the upper left hand corner of the area which will be written in FireCracker 24 memory. This allows you to place an image into an arbitrary place in the FireCracker 24 display. For example, to center a 320 by 200 pixel image on the FireCracker 24 display (when the FireCracker 24 is set to 768 pixels wide) you would set the **Dest Offset X** to 224 and the **Dest Offset Y** to 141.²

When scrolling is active, you can set the size of the basic scroll by placing values in the **Scroll Size** settings. These values determine how many pixels to the left or right (the **X** setting) or up and down (the **Y** setting) the image will move when the cursor keys are struck.

The **Scroll Active** button is necessary to tell the saver that you intend to use the cursor keys for scrolling when this control panel is displayed. Otherwise, the cursor keys will be used to move the cursor in the currently active input field. To use the cursor keys to scroll, select the button marked **Scroll Active** so that it remains highlighted. Now, depressing the cursor keys will cause the image on the FireCracker 24 to scroll in the specified direction. Note that scrolling is always enabled when the previous control panel (the one shown in Figure 7.6) is displayed. This is because there are no input fields in that control panel, allowing the cursor keys to be used for scrolling purposes.

Note that the scrolling takes place only within the window you specified using the **Dest Size** and **Offset** fields.

²This is $(768 - 320)/2$ for the x offset and $(482 - 200)/2$ for the y offset.

If you depress one of the cursor keys while the **Shift** key is also depressed, you will scroll three times the distance specified in the **Scroll Size** input fields. If you hold down the **Ctrl** key at the same time as one of the cursor keys, you will scroll to the far edge in the direction you specified.

When Amiga video is disabled, scrolling is automatically enabled.

Selecting the button marked **Previous Panel** will return you to the panel shown in Figure 7.6.

7.9 FRAMEBUFFER

As of Art Department Professional 2.3.0 the Mimetics FRAMEBUFFER saver module has been taken out of the package. This was done after serious consideration to the current state (availability) of the device. Owners of previous versions of Art Department Professional should install this version of Art Department Professional into the same directory as their current version—the FRAMEBUFFER saver module has not been modified for a long time. New owners of Art Department Professional who need this module should contact Technical Support.

7.10 FRAMESTORE

The FRAMESTORE saver allows you to export the raw image data currently in ADPro's image buffer in Framestore format, which can then be read directly by the Video Toaster 2.0 (or later) software. No Toaster hardware need be present nor active for this operation to work.

Only 24-bit information will be saved out. If raw (8-bit) gray image data is present, then it will be padded to 24 bits as it is being saved. This is a convenience so you do not have to execute the **Gray_To_Color** operator before saving it out. If you then reload the saved file, the Image Information area on the main window of ADPro will display **Color** image data, although the rendering will still appear in shades of gray. In this case, however, you should use the **Color_To_Gray** operator to make sure that within ADPro it is gray image data—the operations will be more accurate if grayscale images are represented in ADPro as gray image data.

Upon invoking the saver, its control panel (shown in Figure 7.8) will appear. It is from this panel where you can specify how to save the Framestore file.

The **Filter:** slider and input field control what type of filtering will be performed on the image, if at all. You can select one of the five possible choices:

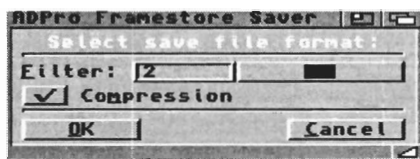


Figure 7.8: The FRAMESTORE saver control panel.

0 through 4. If you want a quick preview (during a Framestore load from the Toaster software) of your image, then select 0. This setting has the highest probability of producing dot crawl, but produces the sharpest images. To reduce the amount of dot crawl but increase the probability of color bleeds, select one of the other options. The 4 setting produces the least or no dot crawl at all, but may bleed the most.

The time it takes to save out the Framestore using the 1 through 4 settings will be comparable to each other, but a **Filter:** setting of 0 will take the least amount of time since no filtering needs to be performed. There is no discernible difference in the loading times among these modes.

The best choice for you depends upon your needs. If you simply want a quick preview of your images to check for general correctness of the image, then use 0. If, however, you want the highest quality image that can be produced by the saver module but do not have a strict need to load and display the image quickly, then select 4. Otherwise, one of the middle settings should be adequate enough for normal use. Setting this control to 2 will give you an image equivalent to what Toaster Paint will give you.

The **Compression:** checkbox allows you to specify whether the image will be saved in compressed (checked) or uncompressed (not checked) Framestore format. Compressed Framestores, as the name implies, doesn't take up as much hard drive space as uncompressed ones.

If you press the **OK** button, a file requester will appear asking for you to specify a filename for the Framestore file to save. Once specified, it will attempt to save the file. The meter window will appear, showing the progress of the operation. Once the meter's "needle" completes its pass, you can continue with any other operation.

If you press the **Cancel** button, the no image data will be saved, and control will return to the main screen.

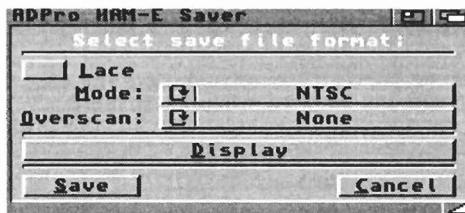


Figure 7.9: The HAME Saver control panel.

7.11 GIF

The GIF saver requires the presence of rendered image data which cannot be in the Amiga specific formats of HAM (Hold-And-Modify) or EHB (Extra-Halfbrite).

Upon selecting the GIF saver, you will be given the choice of which data to save. If the rendered data is in one of the allowable Amiga display modes, you can select **Cropped Screen Image** to save only the portion of the image which would be visible if you selected the **Redisplay** button. Or, you can select the button describing the full size and number of colors of the rendered data to save the entire rendered image.

The file requester will then appear. Use it to select a file name in which to store the GIF image.

Note that you can benefit from use of the enhanced palette mode when saving images for use with applications which can read GIF.

7.12 HAME

The HAME saver (whose control panel is shown in Figure 7.9) gives you control over the HAME display device from Black Belt Systems. The HAME works in two modes:

1. In register mode, the HAME displays an image with 2 to 256 palette mapped colors. To produce a register mode image for use with the HAME saver, simply produce a rendering having between 2 and 256 colors and invoke the HAME saver. Use of the enhanced palette mode is suggested.
2. In HAME mode, the HAME displays an image using the same sort of tricks that the Amiga's own HAM mode employs to gain the impression

of higher color content without greatly increasing the memory required to display such an image.

HAME mode is supported in both ADPro's HAM mode as well as ADPro's HAM8 mode. When used in conjunction with renderings performed in ADPro's HAM8 mode, HAME mode can show colors from a 262,144 color palette. In either case, the use of the enhanced palette mode is suggested.

Creating a HAME mode rendering in ADPro's HAM mode is quite simple. Just perform the following steps:

1. Select the HAM rendering mode in ADPro and render an image as you would normally.
2. Select the HAME saver and use it to either display or save an image in the HAME format.

Creating a HAME mode rendering in ADPro's HAM8 mode is slightly more complex. Follow these steps:

1. Enter the Palette control panel and select the **Custom Palette** checkbox so that it is "on".
2. Select HAM8 for **Colors (Total):**.
3. Select some number less than or equal to 60 for **Colors (Used):**. At 60 colors used, image quality will be maximized.
4. Ensure that **Offset Color Zero:** is set to 0.
5. You are encouraged to utilize the enhanced palette mode.
6. Accept these settings on the Palette control panel.
7. Select the desired screen settings, such as interlaced or overscan. Render the image by pressing the **Execute** button.
8. Select the HAME saver and use it to either display or save an image in the HAME format.

Upon entering, the HAME saver automatically detects if the image data is rendered in a palette mapped mode (2 to 256 colors) or in a HAM mode (HAM or HAM8). If in a palette mapped mode, the HAME saver automatically reformats the rendered image to create a register mode image. If the rendered image data is in a HAM mode, the HAME saver automatically reformats the rendered image data into a HAME mode image.

The HAME saver will inform you which type of image it has created via a text message contained in the saver's control panel. The control panel gives you the choice of displaying or saving the HAME image. If you elect to display the

image, it will be displayed until you click the left mouse button. Note that unlike elsewhere in ADPro, you cannot scroll this display. If you elect to save the image, a file requester will appear for you to select the name under which the image should be saved.

The **Overscan**: cycle gadget on the HAME control panel allows you to set the type of overscan the image will be displayed with. **Width** will display only horizontal overscan. **Height** will display only vertical overscan. Finally, **Width & Height** will display both vertical and horizontal overscan. When the **Overscan**: gadget is set to simply **None**, the image will be displayed with no overscan at all.

NOTE The HAME device is a *low resolution* device. That is, while the saver pays attention to the settings defining overscan, interlace, and choice of NTSC or PAL, it does not pay attention to the button choosing between high and low resolution.

7.13 HARLEQUIN

The HARLEQUIN saver gives you control over the display board of the same name from Amiga Centre Scotland. There are several Harlequin versions with differing capabilities. The HARLEQUIN saver is written to take advantage of the features of the top end version, including the ability to double buffer, genlock, and use an alpha channel.

Figure 7.10 shows the control panel for the HARLEQUIN saver. If you have more than one Harlequin installed in your system, the button marked **Board #** allows you to select which board you will be controlling at the moment. The button marked **Width** allows you to cycle through the available display widths that the Harlequin supports. Note that if your board has a double buffering capability, changes to the width affect both buffers.

If your board supports double buffering, you can select which buffer will be written to by the next **Image** command with the button marked **Image To**. The button marked **A** to the right of the **Image To** button will become unghosted if you are currently working on grayscale data and your Harlequin has an alpha channel capability.

Assuming your board does, and you are currently working on a grayscale image, you can either render the image into the 24 bit-plane buffer or into the 8 bit-plane alpha channel. Selecting the button marked **A** causes the next **Image** command to write to the alpha channel rather than the main buffer.

The button marked **Display** allows you to select which 24 bit-plane buffer the Harlequin will display from if it is equipped for double buffering. The button



Figure 7.10: The ACS HARLEQUIN saver control panel.

marked **G** turns the genlocking capability of the Harlequin on or off for the currently displayed buffer. The button marked **A** turns the display of an alpha channel on or off for the currently displayed buffer, if your board is equipped with alpha channel capability.

To summarize, you can image to either buffer 0, buffer 1, the alpha channel for buffer 0, or the alpha channel for buffer 1. You can display from buffer 0 or 1, with or without genlock, and/or alpha channel.

The **Src Offset** input field represents where within your image the saver will begin fetching image data to send to the Harlequin. The upper left hand corner of your image is (0,0). The **Dest Size** defines the size of the rectangle (the "destination rectangle") of image data which will be sent to the Harlequin. In general, these values will be set to the size of the Harlequin's screen. However, you can set these values smaller to create a scrolling window on the Harlequin screen.

The **Dest Offset** defines where the destination rectangle will be placed on the Harlequin screen. In general this will be set to (0,0).

The **Scroll Size** values determine how many rows or columns are shifted when scrolling is enabled. Scrolling is enabled when the **Scroll** button is selected. When enabled, the four arrow keys on your keyboard control scrolling. Depressing the left or right arrow key will scroll by the amount specified in the **Scroll Size X** gadget. Depressing the up or down keys will scroll by the amount specified by the **Scroll Size Y** gadget. Holding down the **Shift** key in conjunction with any of the arrow keys triples the corresponding scroll amount. Holding down the **Ctrl** key in conjunction with any of the arrows causes the image to scroll as far as possible in that direction. Again, scrolling is only enabled when the **Scroll** button is selected.

The button marked **Center** causes the offsets and sizes to be adjusted so that the image currently in ADPro's memory will appear centered on the Harlequin screen (at the currently selected width). The button marked **Clear** causes the

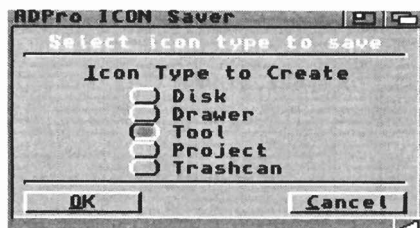


Figure 7.11: The ICON saver control panel.

screen selected with the **Image To** button to be completely cleared.

If raw and rendered image data is available, you can image either by selecting the appropriate setting on the button marked **RAW** in Figure 7.10. The rocker switch marked **Non Lace** toggles the Harlequin into and out of interlaced video mode. The **SCRN ON** rocker switch toggles on or off the Harlequin video output.

The **Quit** button causes the saver to exit and return control to the ADPro main screen. The button marked **Image** causes image data to be sent to the Harlequin subject to the values defined for offsets and sizes.

7.14 ICON

NOTE The ICON saver was developed by Steve Tibbett, and not ASDG, Incorporated. As such, you should contact Steve Tibbett's BBS at (613) 731-3419 for technical support. This module is included in the ADPro package under agreement with Steve Tibbett.

The ICON saver allows you to export the rendered image data within ADPro's image buffer as an Amiga icon (**.info**) file.

Upon launching the saver, the ICON control panel (as shown in Figure 7.11) will appear. In it, you can specify the type of icon to create. Note that the icon that is created will not have an alternate image (i.e., it will use the "complement" of the image).

Select the **OK** to specify a name for the icon, or **Cancel** to exit the saver without saving an icon. The name that you specify should not contain the **.info** extension—the saver will automatically append it.

Using the ICON saver instead of the CLIPBOARD saver and IconEdit combination allows you to create icons which are larger than the *Amiga® User Interface Style Guide* recommended 80 pixels wide by 40 pixel high.

7.15 IFF

The IFF saver allows you to save the current raw, rendered, or screen image data in IFF-ILBM format. You will be asked to select one of the following:

- **Cropped Screen Image** — Selecting this mode will save the screen image data which would be displayed if you select the **Redisplay** button. This choice is available only if you have rendered in an Amiga displayable mode.
- ***n* Color Image** — Selecting this mode will save the entire rendered image. The number of colors in the rendered image will be presented in place of the *n*. This choice is available whenever rendered image data is available even if the rendering mode is not directly Amiga displayable.
- ***n* Bit-Plane Raw Image** — Selecting this mode will save the entire raw image as either a 24 bit-plane color image or 8 bit-plane grayscale image.

Whenever a raw image is saved, ADPro includes Commodore standard **CLUT** chunks which represent your current color balancing settings. However, many programs which now support Commodore standard 24 bit-plane IFF files do not yet support **CLUT** chunks. To enable you to get exactly the image you intend, you can use the **Apply_Map** to permanently modify the raw data to include the effect of the **CLUT** chunks.

Raw images saved in the IFF format by ADPro also include an **ASDG** chunk which preserves the internal representation of your current color balancing settings. Reading and rewriting such a file in a non-ASDG program will more than likely cause the **ASDG** chunk to be removed. However, this means only that the color balance settings are lost and not that the data has been modified. You can reset the color balance settings by hand if you need them again.

After selecting which type of data to be saved by the IFF saver, the file requester will appear allowing you to specify the output file destination.

The IFF saver will now preserve the screen mode settings defined either at the time the last image was loaded or at the time you pressed the **Execute** button to create rendered image data. This modification was done to compensate for the inability of some programs to handle the IFF files saved from previous versions of ADPro.

7.16 IMPULSE

The IMPULSE saver requires the presence of 24 bit-plane raw color image data.

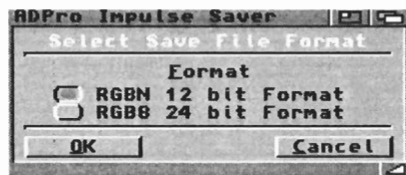


Figure 7.12: The IMPULSE saver control panel.

You will be asked to select between RGBN and RGB8 formats. The RGBN format is a 12 bit-plane format in which ADPro will truncate the least significant four bits from each of the red, green, and blue components of the image. The RGB8 format is a 24 bit-plane format in which ADPro will preserve the entire raw image it has.

After selecting which type of data to be saved by the IMPULSE saver, the file requester will appear allowing you to specify the output file destination.

While this format supports only raw image data, it does not contain its own color look-up tables. Therefore, in order to preserve the effect of any color balance settings you might wish to maintain, use the Apply_Map operator to transfer the effect of the color settings into the raw image data.

7.17 IV24

The IV24 saver gives you control over the GVP Impact Vision 24's display capabilities.

NOTE You must have an IV24 installed in your machine and you must have installed the `fye.library` in your `LIBS:` directory to make use of this saver. If these conditions are not met, selecting this saver will cause an error message to be displayed on ADPro's main screen.

Figure 7.13 shows the control panel which will appear when you execute the IV24 saver.

The **X** and **Y** values of **Src Offset** allow you to select the upper left hand corner in the source image data (the image in ADPro memory) from which to start transferring image data to the IV24. In general, when an image is smaller than the currently selected IV24 display size, these values will be set to zero. Setting these values to non-zero has the effect of "cropping" the image actually sent to the IV24 without actually disturbing the image data in ADPro's memory.

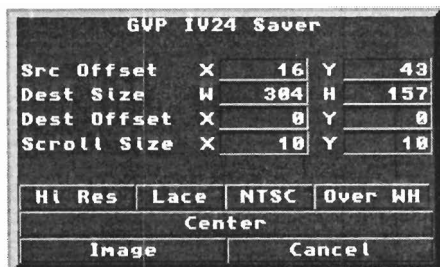


Figure 7.13: The IV24 saver control panel.

The **Dest Size** values determine how large an area on the IV24 will be written into. This also determines how large an area from the source image (the image in ADPro memory) will be accessed. In the example of Figure 7.13, selecting the **Image** button will transfer a 304 by 157 pixel area from ADPro memory.

The **Dest Offset** values determine the upper left hand corner of the area which will be written in IV24 memory. This allows you to place an image into an arbitrary place in the IV24 display.

Selecting the button marked **Image** causes image data to be transferred to the IV24. The size of the screen opened for the IV24 depends upon the settings of the four buttons located above the button marked **Center**.

You can open the IV24 with a high or low resolution screen, in interlace or not, in the NTSC or PAL video mode, or with overscan.

After pressing the **Image** button, you can use the cursor keys to scroll about the image. You can set the size of the basic scroll by placing values in the **Scroll Size** settings. These values determine how many pixels to the left or right (the **X** setting) or up and down (the **Y** setting) the image will move when the cursor keys are struck.

When viewing the image on the IV24, pressing the cursor keys will cause the image to scroll in the specified direction. Note that the scrolling takes place only within the window you specified using the **Dest Size** and **Offset** fields.

If you depress one of the cursor keys while the **Shift** key is also depressed, you will scroll three times the distance specified in the **Scroll Size** input fields. If you hold down the **Ctrl** key at the same time as one of the cursor keys, you will scroll to the far edge in the direction you specified.

The button marked **Center** will cause your image to be centered on the currently defined IV24 screen. It does this by setting appropriate values in the **Offset** and **Size** input fields.

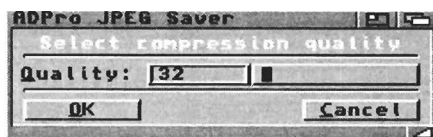


Figure 7.14: The JPEG saver control panel.

7.18 JPEG

The JPEG saver can create files conforming to the JFIF file format standard. JPEG itself stands for the Joint Photographic Expert Group and is an image compression technology which dramatically reduces the size of an image for storage or transmission purposes. JFIF is a standard for constructing JPEG files.

Note that JPEG is a *lossy* compression method. That is, if you compress an image using the JPEG saver and then decompress it using the JPEG loader, the resulting image will not be exactly the same as the image you started with. The JPEG algorithm is designed with the human visual system in mind, so the changes it makes to an image are often unnoticeable. In fact, you can control the amount of change made to an image directly since you can control how much compression will take place. Higher compression means more change to the image.

The JPEG saver control panel, shown in Figure 7.14, allows you to control the amount of compression (which inversely affects image quality) and which type of JPEG encoding will be used. Note that the JPEG saver can save only raw image data.

The quality slider and input field ranges from 1 to 1000. The higher this number is, the less compression will take place (which means that the quality of the resulting will be higher). A reasonable default value of 32 can often produce compression of 40 to 1 or better with little visually detectable loss.

The boosted quality mode is appropriate for when you demand as little loss as possible from the JPEG algorithm. In fact, at quality level of 1000, no loss is contributed by the quantization step of the JPEG algorithm. Amazingly, even with such minimal loss, compression of 10 to 1 is still possible.

Selecting the **OK** button will cause the file requester to appear. You can use the file requester to select the name of the JPEG file which will be written. JPEG files are often named with a suffix of ".jpg".

Note that the JPEG saver cannot save files in the HSI variant, although the JPEG loader can.

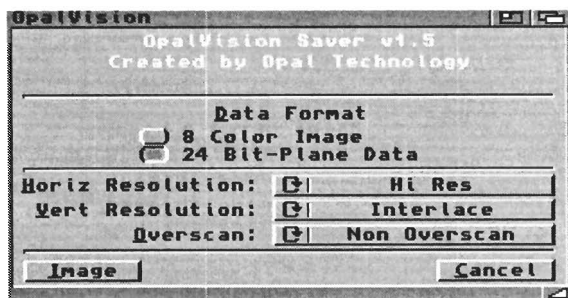


Figure 7.15: The OPALVISION saver control panel.

7.19 OPALVISION

NOTE The OPALVISION saver was developed by Opal Technology, and not ASDG, Incorporated. As such, you should contact Opal Technology for technical support. This module is included in the ADPro package under agreement with Opal Technology.

NOTE This saver module is similar to the OpalVision display module, except that this requires that you switch your saver module. If you want a quick method of displaying an image to the OpalVision graphics board, you might want to use the display module support instead. See Section 9.4.1 for more information on using this display module. This saver module is also included for compatibility with ARexx scripts which require the presence of an OPALVISION saver module.

You must have version 1.6 or later of `opal.library` installed in your `LIBS:` directory. If you find that the saver crashes and you are running under Workbench 2.0 or later of the Amiga's operating system, type the following command at a Shell prompt to determine the version installed:

```
version LIBS:opal.library full
```

The OPALVISION saver gives you control over the OpalVision display board from within ADPro. The saver allows raw 24 bit color or grayscale images or rendered images from 2 to 256 colors to be displayed.

When the saver is activated, the OPALVISION saver control panel (as shown in Figure 7.15) is displayed. This control panel contains several buttons to control the image being displayed.

Key Sequence	Function
Del	Toggle cross hair pointer on/off
Arrow keys	Scroll display
Shift + Arrow keys	Scroll display in larger steps
Alt + Arrow keys	Jump to the extremities of the image
n	Center screen under mouse pointer (if possible)

Table 7.2: The OPALVISION saver keyboard equivalents.

If there is only raw image data, only three cycle gadgets will appear. If there is both raw and rendered data, then two additional radio buttons will appear above these cycle gadgets to let you select which type of data to display.

These radio buttons allows you to toggle between the raw 24 bit color or grayscale image and the rendered image, if one exists. The rendered image data can contain 1 to 8 bit-planes (2 to 256 colors) in a standard palette mapped configuration. That is, the rendered image data cannot be in one of the rendering modes unique to the Amiga such as HAM, HAM8, or EHB.

The **Horiz Resolution:** controls the horizontal resolution, **Vert Resolution:** controls vertical resolution, and **Overscan:** controls whether the image will be overscan (both horizontal and vertical) or not.

To view the image using the current settings, select the **Image** button. Once the image has been displayed, the saver will accept the keyboard commands listed in Table 7.2.

The arrow scroll keys are only applicable if the image is larger than the display screen size.

NOTE Unlike what is normally expected, the settings in the Balancing control panel will affect the rendered *as well as the raw data*. This means that you do not need to perform an Apply_Map operator to get these settings to affect the displayed raw image data.

To exit the saver, click the left mouse button and you will be returned to ADPro's main screen.

7.20 PCX

The PCX saver can save raw as well as rendered data in both color and grayscale. ZSoft Corporation, the developers of the PCX format, have ex-

tended the PCX specification to allow for 24 bit-plane color images, in addition to images ranging from 1 to 8 bit-planes. Note, however, that some applications may not currently handle the 24 bit-plane format of PCX.

The PCX saver cannot save an image rendered in one of the Amiga specific modes, such as HAM (Hold-And-Modify) or EHB (Extra-Halfbrite).

Upon selecting the PCX saver, you will be given a choice of which data to save. If the rendered data is in one of the allowable Amiga display modes, you can select **Cropped Screen Image** to save only the portion of the image which would be visible if you selected the **Redisplay** button. Or, you can select the button describing the full size and number of colors of the rendered data to save the entire rendered image. Also, you may elect to save the raw image data.

The file requester will appear. Use it to select a file name in which to store the PCX image.

Note that you can benefit from use of the enhanced palette mode when saving rendered images for use with applications which can read PCX.

NOTE Even though you can save 2 through 128 color rendered images in VGA format, they will all be padded to 256 (VGA) format.

7.21 PICASSO

NOTE The PICASSO saver was developed by Village Tronic Marketing GmbH, and not ASDG, Incorporated. As such, you should contact Village Tronic (or Expert Services, the US distributor of the PICASSO II graphics board) for technical support. This module is included in the ADPro package under agreement with Village Tronic and Expert Services.

NOTE This saver module is similar to the Picasso display module, except that this requires that you switch your saver module. If you want a quick method of displaying an image to the Picasso II graphics board, you might want to use the display module support instead. See Section 9.5 for more information on using this display module. This saver module is also included for compatibility with ARexx scripts which require the presence of a PICASSO saver module.

The PICASSO saver lets you display images on the PICASSO II graphics board. It can be accessed through the main control window or through ARexx.

You can render ADPro's image buffer on the Picasso in HiColor (15 or 16 Bit) or even in TrueColor (24 Bit). The module will initially choose the best

matching display mode, but you can easily select another one through its control window. Smaller images will be centered by the saver module, larger ones can be scrolled with the cursor keys.

Control:

You can close the display and quit the PICASSO saver module by clicking with the left mouse button, or by pressing **Esc**, **Return**, **Spacebar**, or **"q"**. As mentioned above you can scroll through images larger than the display with the cursor keys. Together with **Shift** you scroll 10 pixels at once, and together with **Ctrl** you can even jump over the whole screen width.

NOTE You can't scroll if you use the PICASSO II in its segmented mode.

ADPro control window:

The **Image** button renders the current ADPro buffer on the Picasso in the chosen display mode.

The **Cancel** button aborts the PICASSO saver.

The **Mode:** listview shows the available display modes. Initially, the saver module chooses the best display mode for the current image, but you can select another one if you prefer. Click on the one you want to use, or double-click on one to automatically display to it.

If you have any questions about this module or suggestions for it, feel free to contact Village Tronic at the following address:

Village Tronic Marketing GmbH
Wellweg 95
D-31 157 Sarstedt
GERMANY

or at the following phone numbers and electronic mail address:

Fax: +49 / (0) 50 66 / 70 13 49
BBS: +49 / (0) 50 66 / 70 13 40
EMail: wollny@arkon.adsp.sub.org

7.22 POSTSCRIPT

PostScript is a typesetting and page description language defined (and trademarked) by Adobe Systems Incorporated. ADPro can save a wide range of

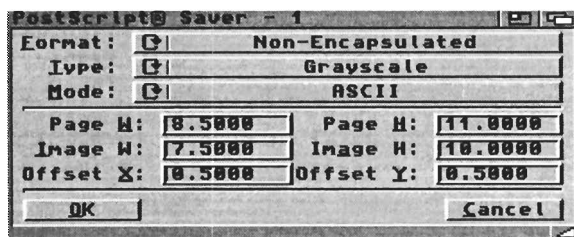


Figure 7.16: The primary POSTSCRIPT saver control panel.

data in the form of a PostScript file. Such a file can be sent directly to a PostScript compatible printer or, in the case of Encapsulated PostScript, can be imported into other programs.

The POSTSCRIPT saver always requires the availability of raw image data. Upon selecting the POSTSCRIPT saver, ADPro will analyze the available raw image data to determine if it is an arbitrary color image, a grayscale image, or contains only shades of one specific primary color (such as cyan, yellow and magenta). This analysis will affect how the POSTSCRIPT saver will compose the final PostScript output.

The POSTSCRIPT saver itself is comprised of three control panels. All three panels will open, with the first (primary) control panel appearing on top of the others. The first of these panels is presented in Figure 7.16.

Panel 1

The format of the PostScript output can be either Encapsulated (EPS) or Non-Encapsulated PostScript. For direct printing purposes, set the **Format:** cycle gadget to **Non-Encapsulated**. For inclusion of the saved PostScript data into other programs, set it to **Encapsulated**.

The type of PostScript data to be saved can be grayscale, color, or separation data. If you set the **Type:** cycle gadget to **Color** PostScript, the resulting file will make use of the PostScript color extensions and can be sent directly to a color PostScript compatible printer. Unless you have a printer capable of supporting color PostScript, the **Grayscale** choice is the appropriate selection. This will allow you to get a grayscale representation of your image right off of your PostScript printer. If you are processing color separation data (ADPro will determine this in the data analysis phase mentioned earlier), you can select a special setting which will include color separation specific instructions in the resulting PostScript file.

The **Mode:** cycle gadget allows you to specify that the image data will be

saved in either **Binary** or **ASCII** form. **Binary** form is much smaller than **ASCII** but can only be used with PostScript printers which are communicating via AppleTalk (registered trademark of Apple Computer) or some other similar printing-capable network. For most ADPro users, the correct setting is **ASCII**.

For non-Encapsulated PostScript output, the following options are available on this first control panel (all measurements are in inches):

- Page width (**Page W:**) and height (**Page H:**) defines the logical size of the PostScript page. Remember that optional features, such as registration and crop marks, exist and are printed outside the logical page. Therefore, you must leave room for them on the physical page should you want them.
- Image width (**Image W:**) and height (**Image H:**) defines the size of the box in which the image will be printed. The actual size of the image is effected by these values as well as whether or not the original aspect of the image will be preserved.
- The x (**Offset X:**) and y (**Offset Y:**) offset of the image within the logical page specifies where the upper left hand corner of the image box will be relative to the upper left hand corner of the logical page.

The goal in designing ADPro's POSTSCRIPT saver was to provide the maximum amount of flexibility to the user. This comes at the cost of some confusion, however, since we provide so many ways of adjusting the output image. The possible confusion may be alleviated somewhat by Figure 7.17.

In Figure 7.17, notice that the registration and crop marks appear outside the logical page. Also note that the amount of overlap of the logical page and the crop marks is determined by the bleed setting. Be sure to set aside space on the physical page (by reducing the size of the logical page) if you wish to be able to see all requested crop and registration marks.

Panel 2

To activate and bring to front any one of the three panels, simply press the 1, 2, or 3 key, which correspond to the panel number (each panel's number is displayed in its title bar).

The second POSTSCRIPT control panel is shown in Figure 7.18. Here, you may specify the angles and densities to be used in the PostScript output file.

The input field marked **C/R Dns:** allows you to specify the Cyan density (all densities are in lines-per-inch) for color separation PostScript files or the Red density in color PostScript files. The input field marked **M/G Dns:** allows you to specify the Magenta density for color separation PostScript files or the Green density in color PostScript files.

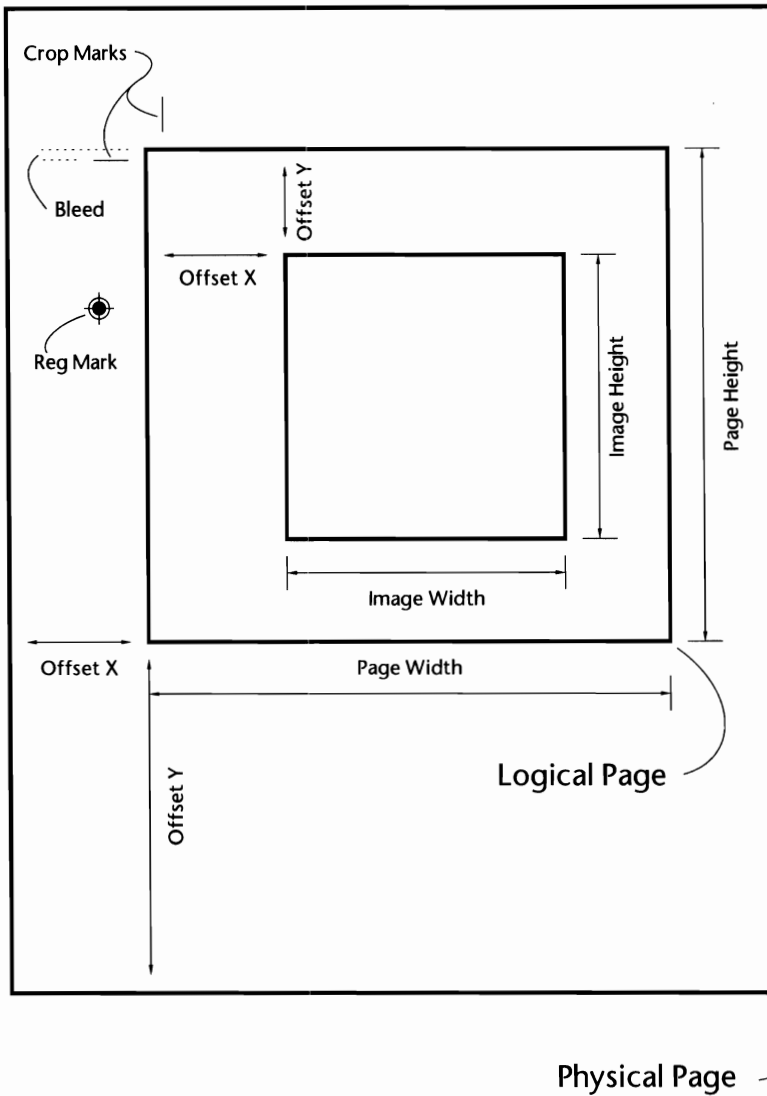


Figure 7.17: Various PostScript metrics.

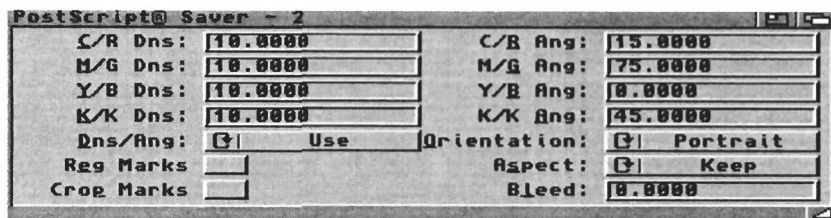


Figure 7.18: The second POSTSCRIPT saver control panel.

Similarly, the input field marked **Y/B Dns:** allows you to specify the Yellow density for color separation PostScript files or the Blue density in color PostScript files. Finally, the input field marked **K/K Dns:** allows you to specify the Black density for grayscale, color separation, and color PostScript files.

NOTE When printing grayscale image data (different from printing a color image in grayscale PostScript) only the black density is used.

The input field marked **C/R Ang:** allows you to specify the Cyan angle (all angles are in degrees) for color separation PostScript files or the Red angle in color PostScript files. The input field marked **M/G Ang:** allows you to specify the Magenta angle for color separation PostScript files or the Green angle in color PostScript files.

Similarly, the input field marked **Y/B Ang:** allows you to specify the Yellow angle for color separation PostScript files or the Blue angle in color PostScript files. Finally, the input field marked **K/K Ang:** allows you to specify the Black angle for grayscale, color separation and color PostScript files.

In Figure 7.18, the cycle gadget labelled **Dns/Ang:** can be toggled between **Use** and **Ignore**. When set to **Use**, the PostScript file will contain instructions telling the printer to use the angles and densities that you have specified. When set to **Ignore**, the values you specify are ignored. Instead, the PostScript file contains instructions telling the printer to use its own internally stored default angles and densities.

You may wish to set this gadget to the **Ignore** setting if you are working with a sophisticated printer which may have finely tuned settings already stored in its firmware.

The **Orientation:** cycle gadget can be toggled between **Portrait** and **Landscape**. Selecting **Landscape** rotates the logical page (see Figure 7.17) by 90 degrees.

Registration marks can be toggled on or off with the **Reg Marks** checkbox.

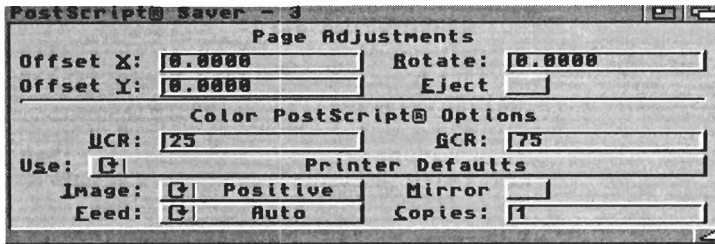


Figure 7.19: The third POSTSCRIPT saver control panel.

Similarly, crop marks can be toggled on or off with the **Crop Marks** checkbox.

Note that crop and registration marks appear outside the logical page (see Figure 7.17). Remember to take this into account when specifying the page dimensions by ensuring that the logical page plus the space set aside for crop and registration marks can fit on the physical page.

The input field marked **Bleed:** controls how much the crop marks (if present) will overlap into the logical page.

The **Aspect:** cycle gadget in Figure 7.18, when set to **Ignore**, causes the image to be stretched as needed to fill the entire image width and height. Switching this gadget to **Keep** causes the image will be stretched to fill only one dimension of the specified image area. The other dimension will be stretched to match the scaling of the first dimension.

Panel 3

The third POSTSCRIPT saver control panel is shown in Figure 7.19.

In this figure, the input fields marked **Offset X:** and **Offset Y:** allow you to specify the offset of the lower left hand corner of the logical page relative to the lower left hand corner of the physical page. The input field marked **Rotate:** allows you to rotate the entire page about its lower left hand corner.

The **Eject** checkbox in Figure 7.19 may be toggled on or off. When off, the PostScript file will contain instructions telling the printer not to eject the page when it is finished rendering inside the printer. This allows you to send additional pages which will be overlayed over the first. If on, the resulting PostScript file contains instructions telling the printer to eject the page as soon as it is completely rendered within the printer.

The **Image:** cycle gadget allows you to specify that you wish a positive (**Positive**) or negative (**Negative**) image rendered on the printer. To create a set of

negative films (used for most printing applications) you would set this button to its **Negative** state. A special feature of the ADPro POSTSCRIPT saver is that it will actually produce a true negative even on PostScript printers which do not support the standard PostScript **negative** operator, such as the Apple LaserWriter series.

The **Mirror** checkbox in conjunction with the **Image:** cycle gadget will allow you to create mirror image negatives which is the standard used by professional printers. A special feature of the ADPro POSTSCRIPT saver is that it will actually produce a true mirror image even on PostScript printers which do not support the standard PostScript **mirror** operator, such as the Apple LaserWriter series.

The **Feed:** cycle gadget, when set to **Auto**, specifies that the PostScript file should contain instructions telling the printer to take paper from its standard paper tray. Toggling this gadget to **Manual** causes the PostScript file to contain instructions telling the printer to insist on a manually fed sheet of paper even if paper is available in its standard paper tray.

Finally, the input field marked **Copies:** allows you to specify the number of copies that the printer will be told to make of each page output.

Main Controls

If the **Cancel** button is selected from the first control panel, then the POSTSCRIPT saver will exit and no image will be saved.

To accept changes made to the control panels, select the button marked **OK** (in the first control panel). The file requester will appear, from which you can select the name of the file into which the PostScript output will be saved.

To output the PostScript instructions to a PostScript printer connected to your Amiga computer through either the internal serial or parallel port, follow the steps below:

1. Blank out the **Drawer** input field. Be sure to press the **Return** key.
2. Insert the name of the assigned device, such as **SER:** for serial or **PAR:** for parallel, in the **File** input field.

NOTE Do NOT press the **Return** key—simply select the **OK** button. Otherwise, the ASL requester will want to put the device name in the **Drawer** field.

Remember to press the **Return** key after changing the contents of the input fields.

7.23 PREFPRINTER

The PREFPRINTER saver allows ADPro to print the currently loaded image on any printer for which a Preferences printer driver (that supports strip printing) exists or to the FARGO Primera dye-sublimation (photo-realistic) printer. If you have a color printer, you can print with 24 bit-planes of color accuracy. If you have a black-and-white-only printer, then you can print with 8 bit-planes of grayscale accuracy. If you were not using ADPro's printing capability, the Amiga's own color printing capacity would only be 12 bit-planes in color and 4 bit-planes in grayscale. Therefore, printing through ADPro allows you much better fidelity than if you had not printed through ADPro.

Also, the PREFPRINTER saver allows you to print poster-size images spread over multiple pages. In fact, ADPro itself does not impose any size limitation on the resulting print.

Selecting the PREFPRINTER saver causes the control screen shown in Figure 7.20 to appear. The screen is dominated by two features. First, in the backdrop, there is a ruled grid which represents the image to be printed, relative to page size and boundaries. Second, in a movable panel, are the many controls and displays that you use to get the printed results you want.

From either the panel or the screen, there are controls in the menu bar. Use the right mouse button to get to them.

7.23.1 Setting the Preferences Paper Type

Using the PREFPRINTER saver, how you set the paper type in the Preferences Printer editor may be important.

If you have a single sheet **only** printer, such as a laser printer, skip this section.

If you have a printer which can take single sheets or continuous paper (such as fan-folded or rolled paper), PREFPRINTER can print multiple pages with either a gap or no gap between the pages. You can affect this by setting the paper type in Preferences to either "Fanfold" or "Single".

If you set "Single", PREFPRINTER will force a form feed between pages.

If you set "Fanfold", PREFPRINTER will not force a form feed, but some printer drivers may insert one themselves.

If you are printing a multi-column poster on continuous paper (fanfold or roll), then you would set Preferences to "Fanfold" and define some top margin (**Print Dimensions Left:** or **Top:** depending upon print orientation) so that the columns of your print will be separated. Also, when printing posters on continuous paper, you should set the **Page Definitions Top:** margin to 0 so

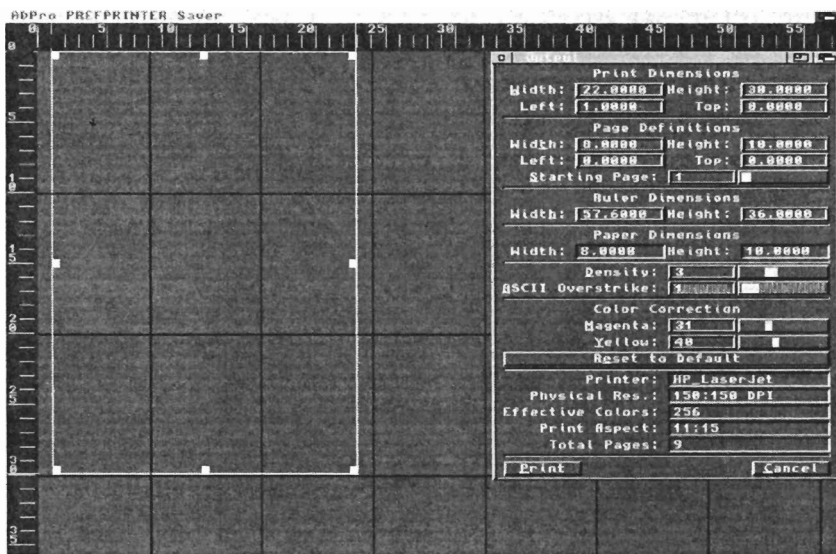


Figure 7.20: The PREFPRINTER control screen. Note the print direction arrow in the middle of the first page to be printed.

that there is no gap between individual pages of a column.

7.23.2 Specifying the Print Destination

PREFPRINTER can output to either the Preferences-supported printer or to the FARGO Primera printer (with dye-sublimation kit installed). Select which to output to using the **Print To** control under the **Project** menu.

Each mode will enable various controls in the Output control panel and menus, while disabling others.

If you only want to initiate a form feed (especially handy for tractor-fed paper), use the **Form Feed** menu item (under the **Output** menu).

7.23.3 Specifying the On-Screen Dimensions

The ruled scale along the top and left edges of your screen automatically size themselves so that they will fill the width and height of the screen. The area depicted along these rulers is determined by several settings in the control panel.

First, the overall dimensions of the depicted area are set using the two **Ruler Dimensions** fields. If the **Measuring System** option (under the **Settings** menu) is set to **English**, then the unit of measure used in the **Ruler Dimensions** is either inches or feet (set in the **Units** submenu). If it is set to **Metric**, then the unit of measure used in the **Ruler Dimensions** is centimeters or decimeters (also set under **Units**).

Changing one of the **Ruler Dimensions** automatically changes the other dimension so that the resulting depiction will fit properly on one Amiga screen.

If you think of the ruled area as a canvas upon which you will place the image to be printed, the **Ruler Dimensions** define how big the canvas is. Making these values large means that you can depict many printer pages on screen at the same time. However, it means you have less visual resolution to make on-screen adjustments to the size and location of the area to be actually printed.

Therefore, if you are printing only a single page, it is best to set the **Ruler Dimensions** to reasonable values, such as 1.5 or 2 times the size of your printer's physical page size.

As indicated above, you can set the unit of measure standard to either the English or Metric system. This can be done by selecting one of them in the **Measuring System** submenu under the **Settings** menu.

7.23.4 Setting the Printer Driver and Its Attributes

Your choice of which printer is currently selected by Preferences will affect the operation of the PREFPRINTER saver in many ways. The PREFPRINTER saver shows you which printer it believes to be the currently selected printer in the text field marked **Printer:**. The PREFPRINTER saver asks Preferences how many dots-per-inch the currently selected printer is capable of and displays this information on the next line, labeled **Physical Res.:** (for physical resolution).

Note that many Amiga Preferences printer drivers allow you to select several different printer densities. Selecting a different density from Preferences will be reflected by the PREFPRINTER saver. You can also set the printer's density directly from within the PREFPRINTER saver with the input field and slider marked **Density:**.

Also note that the PREFPRINTER saver interrogates Preferences to determine which printer to use (and other printer specific information) at the time the saver starts up. If you change any Preferences printer settings while the PREFPRINTER saver is running, these changes will not take affect until the next time you enter the saver.

Mask Size	Effective Colors	Effective Gray Shades
2	65	5
4	4097	17
8	262145	65
16	16777216	256

Table 7.3: Dither mask size versus effective color resolution.

7.23.5 Setting the Dither Type

Many dithering methods are available from PREFPRINTER. These are described below. In each case a larger dither mask (matrix) size produces a print which can represent a wider range of colors but will produce less spatial information per unit area of paper. Conversely, a smaller dither mask size can reproduce fewer colors (or shades) but more closely approximates the true resolution of your printer.

Another way of expressing this is the following: There is a tradeoff between printing many colors and printing in high resolution. Given a specific printer with a specific DPI capability, asking for many colors means using a larger dither mask size. A larger dither mask size cuts down on your effective resolution. (Table 7.3 shows the relationship between the size of the dither mask and the effective number of colors you will be able to reproduce.)

Note that this can work to your advantage when blowing a picture up in size. Blowing up a picture means that there are more dots to work with, which offsets the loss in resolution caused by a larger dither mask size. This, added to the benefits to be had by being able to reproduce more colors (or shades), means that your enlarged posters will look quite good.

Also note that many inexpensive printers, including most laser printers and dot matrix printers, have considerable dot gain. For example, a 300 DPI laser printer does not actually print dots which are $\frac{1}{300}$ of an inch in size. Rather, its dots will be much larger. This causes some dithers, such as the Floyd-Steinberg and Ordered dithers, to produce intensely over-saturated prints. Other dithers such as the two halftone dithers overcome this problem with low-end printers.

You might expect the values listed in Table 7.3 under the Effective Colors column to be 64, 4096, 262144, and 16777216, and under the Effective Gray Shades column to be 4, 16, 64, and 256. This might be correct for many other programs, but with PREFPRINTER the above is correct.

The best way to visualize the difference is with a Mask Size of 2. Assume you want to print a sample of all 256 possible shades of gray (where each shade has

the same width as the others, and all are placed next to each other from darkest to lightest shade). With many other printing programs, this gray “ramp” would not print as 5 equal-width shades of gray, but with PREFPRINTER it will. This will tell you that all 5 possible shades (for the example Mask Size of 2) are being used correctly.

To select a dither type, choose it from the **Dither** submenu under the **Output** menu. To select a mask size (for those dithers which use it), choose it from the **Mask Size** submenu (also under **Output**).

The specific dithers available through PREFPRINTER include:

Floyd The Floyd (short for Floyd-Steinberg) dither is an error diffusion dither which can produce very good results on color printers with little dot gain and very good registration. If it produces a washed out print or particularly bad patterns, then try another dither.

Ordered The Ordered dither produces a regular repeating pattern which is often used for printing computer graphics. The Ordered dither is particularly susceptible to over-saturation due to dot gain on inexpensive printers.

Horizontal, Vertical, Forward Diagonal, Backward Diagonal

The Horizontal, Vertical, Forward Diagonal, and Backward Diagonal dithers overcome many of the dot gain problems that the Floyd and Ordered dithers have with inexpensive printers. These dithers (especially the Diagonal dithers) are especially good for large posters.

Forward Brick, Backward Brick The Forward Brick and Backward Brick dithers are close cousins of the line-oriented dithers described immediately above. They produce an interesting effect.

Halftone A, Halftone B The Halftone dithers (Halftone A and Halftone B) differ in how they place a halftone matrix.

Halftone A causes the halftone matrix for each of the primary colors to be centered about the same point. This means that the primary colors will overlap completely, leaving a lot of white paper showing through. This may be appropriate for some better dye sublimation type printers or other color printers with good registration where the inks mix well.

Halftone B, on the other hand, staggers the halftone matrix of each primary color so that they do not overlap. This is similar in concept to traditional color offset printing. Halftone B may produce better results on printers whose inks do not mix well, and on printers with less than perfect registration.

The Halftone dithers can produce some extremely good results and compensate for the dot gain problems outlined above. Try both Halftone dithers to see which one is better for your particular printer.

ASCII The ASCII dither is a unique dithering method in that it does not require the printer to be graphics-capable (able to output graphics).

Instead of using a pattern of dots, it uses actual ASCII characters. Although this method will not produce the most accurate images, it does produce acceptable results, especially when viewed from a distance.

Using the ASCII dither also allows owners of simple daisy-wheel printers to output their images to these types of printers, assuming a Preferences printer driver exists for it.

Note that the ASCII dither ignores the **Print Dimensions Left:** value. This means that all of your ASCII printouts will start from the leftmost portion of each sheet of paper.

See Appendix D for samples of each of the above dither types.

If you select the ASCII dither, the **Density:** control will become disabled and the **ASCII Overstrike:** control enabled. This slider allows you to specify how many times to “strike” each line of characters. The more strikes, the closer the printout represents the original image.

7.23.6 Setting the Print Aspect

Below the display of the effective number of colors can be found a display of the current print aspect. In Figure 7.20, the current aspect is given as 11:15. This means that the width of the area which will be printed is eleven fifteenths the size of the height of the area to be printed. This aspect is changed whenever you change the size of the select box on the backdrop, or change the width or height given under the **Print Dimensions** heading.

The aspect of the printed area can also be set correctly automatically by the PREFPRINTER saver by selecting **Set To Image Aspect** in the **Print Aspect** submenu under the **Settings** menu. This command causes the saver to interrogate the pixel aspect defined for both the image within ADPro’s memory and the pixel aspect defined for the printer. It automatically computes some internal ratios so that a circle would print like a circle and a square would be square, regardless of the physical aspect of the printer’s dots. *As long as ADPro’s idea of the image’s pixel aspect is correct*, selecting this command will allow you to print the image correctly, automatically.

Selecting the **Set To Image Aspect** command forces the aspect to be locked (as such, the **Locked** option in the same **Print Aspect** submenu is checked). When this option is checked (selected), any change to the width or height of the select box will be matched with a corresponding change to the other dimension.

So, the quickest way to get a correctly proportioned print in the size of your choosing is to first select **Set To Image Aspect** which automatically switches you into **Locked** mode, then drag out the select box to precisely the size print

you want (or type in either the width or height you want and the PREF-PRINTER saver will compute the appropriate value for the other dimension).

Remember, that the PREFPRINTER saver makes use of ADPro's currently defined internal values for pixel aspect and resolution. If these values are set inappropriately, the PREFPRINTER saver will be unable to correctly compensate for pixel versus printer aspect for you. For more information about setting pixel aspect and image resolution, please refer to the Define.Pxl.Aspect operator..

7.23.7 Specifying the Output Settings

Most of the output controls are available in the Output control panel. It is initially open, but you can temporarily close it to show more of the background screen by either clicking on the panel's close gadget or selecting the **Close** menu item (under the **Output** menu). You can reopen the panel by selecting the **Open...** menu item under the same menu. If you need to see more of the screen but don't want to close the window, you can minimize it by pressing the **Right-Amiga + I** key sequence, the same one available from the main ADPro window to minimize that window. Pressing it again maximizes it to its full size.

The width listed under **Paper Dimensions** is set automatically when the PREFPRINTER saver interrogates Preferences. You must use Preferences to adjust this value. You can specify the physical paper height using the input field. If you are using rolled paper, you can set the physical paper height to any reasonable value. If you are using sheet fed or folded paper, setting the paper height to larger than the physical height simply causes the print to extend over multiple pages. However, we suggest you enter the correct value for physical paper height so that the PREFPRINTER saver can accurately keep track of the number of pages being printed.

Now let's turn to the various controls which affect print size. Figure 7.21 shows the various page set-up dimensions that affect your print.

First, there are the margins physically required by your printer. You cannot modify these and they will be present on every page printed by the PREF-PRINTER saver (including posters).

Subtracting any margins which might be required by your printer from the physical paper size yields the maximum printable area. These are presented to you by the width and height fields below the **Paper Dimensions** label. The PREFPRINTER saver does not allow you to change the paper width, although this can sometimes be done from Preferences. If you do change the paper width from Preferences, remember that you have to exit the PREFPRINTER saver and reenter it for those changes to be visible.

Unlike the physical page width, you can set the physical page height from within the PREFPRINTER saver. This was discussed previously.

Next come the margins you specify with **Page Definition Top:** and **Page Definition Left:**. You might want to use these fields to restrict the printable area to allow you to use narrow paper, for example, on a wide printer. In general, you can leave these fields set to 0 (and leave your **Print Definition Width:** and **Height:** set to the full size of the physically accessible page), which means that your print can occupy the total area physically possible.

Page Definition settings are applied to every page in a poster print.

Finally, the actual size of your print will be determined by the **Print Dimensions Width:** and **Height:** values. The **Print Dimensions Left:** will be applied to all of the left-most pages in a poster print. All of the top most pages of a poster print will be affected by the **Print Dimensions Top:** setting.

The total number of pages which will be printed is given at the bottom of the control panel with the **Total Pages:** label. Remember that PREFPRINTER's final idea of the page's dimensions, is determined by the settings under the **Page Definitions** heading.

You can rotate your print either horizontally or vertically to make it better fit the physical size of your paper. This is done with the **Orientation** control under the **Output** menu. This can often dramatically decrease the number of printed pages when printing posters.

Note that the setting specifying either horizontal or vertical printing will affect the order in which pages of a poster are printed. The first page to be printed will always be the one in the upper left hand corner of the bounding box of the poster. To assist you in determining the order the rest of the pages will be printed, a small arrow (indicating print direction) is printed in the center of the first page which will be printed.

You can set the first page to be printed with the **Starting Page:** control. Being able to do this is an essential feature for those doing poster prints, as printers can often jam or run out of ink in the middle of a poster. To make effective use of this feature, count the number of good pages you have printed. The page you want to begin at is one more than the count of good pages.

If you are using the ASCII dither, you can specify how many times each line of characters will be printed or "struck" using the **ASCII Overwrite:** control. The more times each line is struck, the better it can represent the original image.

7.23.8 Using the Color Correction Controls

Available near the bottom of the control panel are ink compensation controls which allow truer color printing. This allows blues, for example, to be printed as blue rather than as purple. The usage of the color correction gadgets are similar to those found in the color separation control panel. You can disable color correction by setting both the **Magenta:** and **Yellow:** values to 0. A button is provided to restore the default values recommended by ASDG.

7.23.9 Controlling the FARGO Primera Dye-Sublimation Printer

If you've selected to print to the Primera, the **Primera** menu's commands become enabled.

The **Paper** setting specifies the type of paper being printed on.

The **Dither** setting specifies whether dithering will be performed or not. If set to **On**, then it will output 24-bit color (more than 16 million colors). If set to **Off**, then it will only output 12-bit (4,096 colors). You usually want this on. If the image is already 12-bits or is already dithered, you probably would turn this off.

The **Spool Directory...** setting is where the saver will output the temporary files it needs for output. You should select a directory with at most 6 megabytes of free space. By default, this will be set to your **T:** assigned directory (normally assigned to the **T** directory in your RAM disk), but if you have less than 6 megabytes *after running ADPro*, then you should select a location on your hard drive. You might want to consider a location which doesn't get used much, such as the **Trashcan** directory on one of your partitions. The files that are placed in this directory will be deleted after the image is printed.

You can initiate the printer's self-test with the **Self Test...** command. This can come in handy to confirm that the printer is working properly.

The **Heat Setting (n)...** command displays a requester with a slider that controls how dark (high values) or light (low values) the print will be. The default or normal value should be fine for most cases, but you can change it if you feel the print needs it. The number that you select will be displayed in the menu item.

Any Margins Physically Required By The Printer

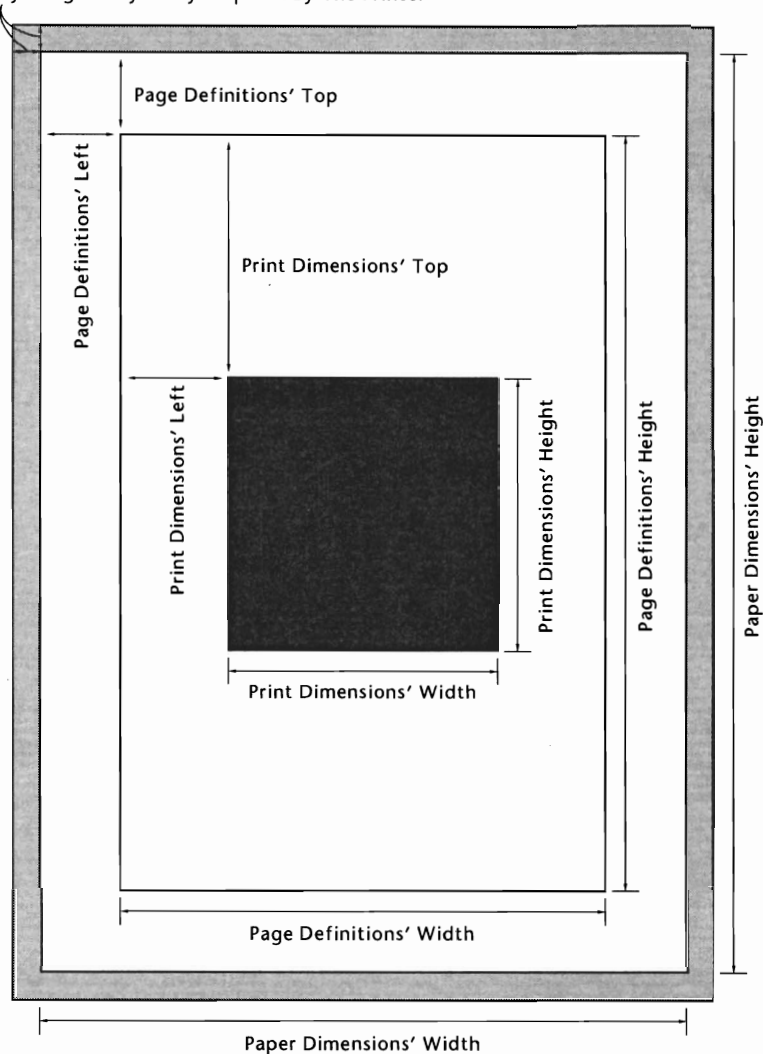


Figure 7.21: Various margins and measures associated with the PREF-PRINTER saver.

7.23.10 Initiating a Print

Selecting the **Print** button or menu item (under the **Project** menu) causes a print to begin. The **Cancel** button or **Quit** menu item (also under **Project**) exits the PREFPRINTER saver without printing the image.

7.24 QRT

The QRT saver requires the presence of 24 bit-plane raw image data. The file requester will appear. Use it to select the name of the file to be written in the QRT format. For more information about the QRT format, see Section 6.29.

7.25 RESOLVER

The RESOLVER saver gives you the ability to directly control the Resolver display board from DMI. This board provides a high quality 8 bit-plane RGB video framebuffer.

The Resolver display board requires the presence of rendered image data containing from 1 to 8 bit-planes in a standard palette mapped configuration. That is, the rendered image data cannot be in one of the rendering modes unique to the Amiga such as HAM (Hold-And-Modify) or EHB (Extra-Halfbrite).

This saver is slightly different from some of the other savers compatible with ADPro in that it allows direct and continuous control over a hardware device allowing multiple operations to be performed without exiting the saver. With a saver such as the one for GIF, the saver is invoked and a file gets saved. With the Resolver saver, the same image (or different parts of the same image) can be saved to the Resolver multiple times without leaving the saver.

The Resolver saver control panel is shown in Figure 7.22. Along the top line, the button marked **Board #** allows you to toggle through which of possibly many Resolver's you wish to control right now. If you have only a single Resolver in your system, this button will display a value of 0 only.

The resolution of the Resolver is determined by the configuration set up by the DMI configuration program. The Resolver board is capable of displaying non-interlaced images up to 2048 x 1024 pixels and interlaced images up to 2048 x 2048 pixels.

The **Center** button automatically adjusts all of the sizing and offset controls of the Resolver saver so that the center of the image data will coincide with the center of the Resolver, taking into account the currently selected screen



Figure 7.22: The RESOLVER saver control panel.

resolution.

The button marked **Clear** will cause the entire screen to be cleared to black.

Selecting the button marked **Image** causes rendered image data to be transferred to the Resolver. The amount of data transferred as well as where the image will be placed is controlled by offset and size settings.

The **X** and **Y** values of **Src Offset** allow you to select the upper left hand corner in the source image data (the image in ADPro memory) from which to start transferring image data to the Resolver. In general, when an image is smaller than the current screen resolution, these values will be set to zero as they are in Figure 7.22. Setting these values to non-zero has the effect of “cropping” the image actually sent to the Resolver without actually disturbing the image data in ADPro memory.

The **Dest Size** values determine how large an area on the Resolver will be written into. This also determines how large an area from the source image (the image in ADPro memory) will be accessed. In the example of Figure 7.22, selecting the **Image** button will transfer a 100 by 100 pixel area from ADPro memory into a 100 by 100 pixel area in Resolver memory.

The **Dest Offset** values determine the upper left hand corner of the area which will be written in Resolver memory. This allows you to place an image into an arbitrary place in the Resolver display. For example, to center (by hand rather than using the **Center** button) a 320 by 200 pixel image on the Resolver display (when set to 736 by 480) you would set the **Dest Offset X** to 208 and the **Dest Offset Y** to 140.³

When scrolling is active, you can set the size of the basic scroll by placing values in the **Scroll Size** settings. These values determine how many pixels to the left or right (the **X** setting) or up and down (the **Y** setting) the image will move when the cursor keys are struck.

³This is $(736 - 320)/2$ for the x offset and $(480 - 200)/2$ for the y offset.

The **Scroll Active** button is necessary to tell the saver that you intend to use the cursor keys for scrolling when this control panel is displayed. Otherwise, the cursor keys will be used to move the cursor in the currently active input field. To use the cursor keys to scroll, select the button marked **Scroll Active** so that it remains highlighted. Now, depressing the cursor keys will cause the image on the Resolver to scroll in the specified direction. The area that will scroll is defined by **Dest Size**.

Note that the scrolling takes place only within the window you specified using the **Dest Size** and **Offset** fields.

If you depress one of the cursor keys while the **Shift** key is also depressed, you will scroll three times the distance specified in the **Scroll Size** input fields. If you hold down the **Ctrl** key at the same time as one of the cursor keys, you will scroll to the far edge in the direction you specified.

7.26 RETINA

NOTE The RETINA saver was developed by MacroSystem, and not ASDG, Incorporated. As such, you should contact MacroSystem for technical support. This module is included in the ADPro package under agreement with MacroSystem.

NOTE This saver module is similar to the Retina display module, except that this requires that you switch your saver module. If you want a quick method of displaying an image to the Retina/Retina Z-III graphics board, you might want to use the display module support instead. See Section 9.6 for more information on using this display module. This saver module is also included for compatibility with ARexx scripts which require the presence of a RETINA saver module.

You must have version 4.x or later of **retina.library** installed in your LIBS: drawer. You may check your version of the **retina.library** by typing the following at a Shell prompt:

```
version LIBS:retina.library full
```

The Retina requires that you have Kickstart v37.175 and Workbench 37.67 or higher.

The Retina saver gives you control over the RetinaTM and Retina Z-IIITM display board from within ADPro. The saver allows 24-bit color, 8-bit grey scale and rendered images to be displayed.

When you select the Retina saver, you will be presented with the option of displaying a rendered image, 8-bit grey scale or 24-bit. The saver has independent buttons for each of the three display options. Rendered or 8-bit data must be available to use these saver options. The display resolutions is defined by your current RetinaScreenMode setting. If the image is larger than the resolution you have set in RetinaScreenMode, you may use the cursor keys to scroll the screen. Consult your Retina user manual for information on changing RetinaScreenMode. Amiga specific modes such as HAM or HAM8 will be displayed on a 24-bit screen.

Clicking the right mouse button while displaying an image will close that screen. Click the Retina saver screen's **About...** button for version information.

7.27 SCULPT

The SCULPT saver requires the presence of 24 bit-plane raw image data. The file requester will appear three times. Use it to select the names of the red, green, and blue component files. Remember that you must explicitly keep track of the width and height of the image in order to make use of it again in the future.

7.28 SEPARATE

Executing the SEPARATE saver brings up ADPro's color separation control panel, as shown in Figure 7.23. Color separation is the process of turning red, green, and blue data appropriate for computer display into cyan, yellow, magenta and usually black data appropriate for print publication.

In order to color separate, raw color data must be available in memory.

ADPro supports 3 different types of color separations. These are:

- RGB separations allow you to split the current color image into its component red, green, and blue planes. Each selected plane will be written to a separate file.
- Three color separations convert the current color image into cyan, yellow, and magenta planes without provision for a black plane. Three color separations are sometimes less expensive to bring to press but will not produce true gray tones which might be a part of the image.
- Four color separations convert the current color image into cyan, yellow, magenta, and black planes. Four color separations are what are generally

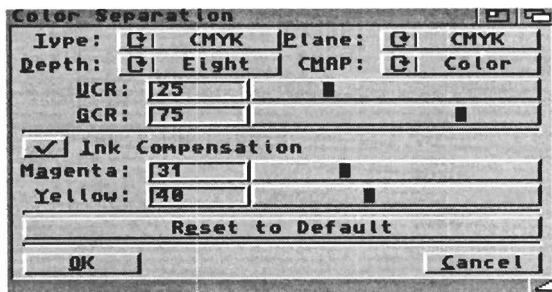


Figure 7.23: The SEPARATE saver control panel.

used in the printing industry today.

Each of these different types of separation can be performed with an accuracy of 24 bit-planes or 12 bit-planes. A 24 bit-plane color separation can represent more than 16 million colors while a 12 bit-plane separation can represent only 4096 different colors.

The disadvantage of 24 bit-plane separations is that they can be quite large (however, 12 bit-plane separations can be quite large as well). The disadvantage of 12 bit-plane separations is that they can represent so few colors.

The choice of which to use (12 or 24 bit-planes) can depend upon many things. Such as:

- If your original image was an Amiga displayable format, the 24 bit-plane color data contains only 4096 colors (however, using ADPro's scaling feature to reduce the size of the image will introduce additional colors to better blend the image). Therefore, you may not get any additional benefit from a 24 bit-plane separation.
- If your image contains many similar hues, such as the slowly changing colors in the human face in sunlight, then you may benefit from the dramatically greater dynamic range of a 24 bit separation. For example, a GIF image with 256 colors may contain a large number of shades of flesh tone. A 12 bit-plane separation can only represent at most 16 similar shades while a 24 bit separation can represent at most 256 similar shades.
- If you plan on using a standard Amiga printer driver and program to output the separations, chances are that the program you will use will not be able to deal with the files a 24 bit-plane separation would produce. However, if you plan on outputting your separation to a PostScript device, you may use either the 12 or 24 bit-plane separations.

Each plane of a 24 bit-plane separation contains 8 bit-planes. Each plane of a 12 bit-plane separation contains 4 bit-planes. You select between the two settings by clicking on the **Depth:** cycle gadget.

Using the **Type:** cycle gadget you can select from among the three types of separation that ADPro supports.

Within each type, you can select all possible planes or any one single plane by toggling the **Plane:** cycle gadget.

At your choice, the color separation files created by ADPro can contain a color or grayscale color map. This choice is determined by the **CMAP:** cycle gadget. When set to **Color**, each color separation file will have an appropriately colored color map included. For example, a cyan separation file will include a color map which ramps from black (where full cyan would be printed on paper) to full cyan (where white would appear on paper).

The **Color** setting is handy in that it allows you to get a visual impression of what a film would look like if printed singly in the appropriate color. And, in order to be compatible with ReSEP (another ASDG product which strips out 12 bit separations produced by older versions of Professional Page, replacing them with 24 bit separations), files must be separated with the **CMAP:** cycle gadget in the **Color** setting.

However, files separated with the **CMAP:** gadget set to **Color** are not technically “correct” separation files in that they contain full intensity cyan (for instance) where there should ultimately be white on paper.

By setting the **CMAP:** gadget to **Gray**, a technically correct grayscale color map will be included with the separation file. When in this setting, white will appear in the separation file where white would appear on paper. Black will appear in the separation file where full colored ink would appear on paper.

Note that separation files created while the **CMAP:** gadget is in the **Gray** setting will not be compatible with ReSEP. However, they will be more appropriate for use with the Inkmun Inks ((714) 948 - 2243) product for the Hewlett-Packard DeskJet 500.

When doing a four color (CYMK) separation, you can control the amount of **UCR** and **GCR** ADPro will perform. UCR (Under Color Removal) defines what percentage of color will be removed in order to compensate for the addition of the black plane. GCR (Gray Component Replacement) defines what percentage of the color removed will be added back in as black.

A four color separation with 0% UCR reverts back into a three color separation (i.e., there’s no accommodation made for black). A four color separation made with 100% UCR will produce a color shifted image very similar in quality to early color photography.

In addition to **UCR:** and **GCR:** controls, ADPro also offers the user complete control over its **Ink Compensation** function. Printer's inks are not completely pure materials. For example, there is some amount of yellow mixed into the magenta ink. And, there is some amount of magenta which unavoidably is found in the cyan ink. The ink compensation values will correct for these impurities.

In general, leave the ink compensation values alone for best results. The ink compensation function can be completely disabled by turning it "off". You will notice that without ADPro's ink compensation function, a blue sky will print as purple. With the ink compensation function set at its default settings, blue skies are blue again.

To restore these default settings at any time, select the **Reset to Default** button.

7.28.1 Using ADPro Separations With Professional Page

Gold Disk's Professional Page (versions prior to 2.0) cannot manipulate images with a higher color resolution than the Amiga currently supports (12 bit-planes or 4096 colors). ASDG developed a program called ReSEP that allows 24 bit-plane separations created by Professional ScanLab, The Art Department, or ADPro to be merged with documents created in Professional Page.

Although the method is slightly circuitous, it works beautifully. Given below is the basic flow of operations:

1. In ADPro, perform any adjustments to the image you wish to make (for example, any color balancing or changes in contrast).
2. Render the image in an any Amiga displayable format and save this rendering to disk. In the steps that follow, we'll refer to this file as the "Amiga displayable" version.
3. Execute a color separation of the same image data and save this to disk. Note that the **CMAP:** cycle gadget must be set to **Color** for separation files to be compatible with ReSEP.
4. In Professional Page, construct the page as you wish placing text and image boxes where you wish them to go. Load the Amiga displayable version of the image (from the steps above) and place this image anywhere on the page as you would with any other Professional Page image box.

You can scale, crop, or cover up this image in any way that you could with any other image box.

5. Execute a color separation from Professional Page, but save the results to disk.
6. In ReSEP, load the PostScript file generated in the previous step by Professional Page. ReSEP will identify all of the image boxes on the page. Using ReSEP select the ADPro separated files to replace the Amiga displayable version of the same image.
When done, press the **Process** button. The output of ReSEP is a second PostScript file identical to the first, except that the selected images have had their 12 bit separations replaced with 24 bit separations.

7.29 TEMP

The TEMP saver allows you to easily and quickly make a “backup” copy of the image currently in ADPro. It uses the image buffer reserved for use by ADPro.

The TEMP saver, in conjunction with the TEMP loader, will allow you to always have a backup copy of your work in progress. Before you make a change which might produce unknown or undesirable results, first save the image using the TEMP saver. If after processing the image you are not satisfied with the results, simply use the TEMP loader to reload the unmodified image. If the image came out the way you intended, then proceed with more changes or resave the new image with the TEMP saver and continue this step-wise process.

Although the TEMP saver can be used for “undo” operations, it can also be used as a way of setting up the use of the same image for manipulation without having to load the original image every single time. See the discussion of the TEMP loader for more details.

Also, the raw image’s pixel aspect is also stored.

There are two ways of determining if image data is currently in the temporary buffer:

1. by selecting **About...** from the **Project** menu to display the information about the program and seeing if an asterisk (*) appears to the right of the **Buffer Size:** value.
2. by scrolling the **Image Information** list text and seeing what percentage of the allocated image buffer is occupied by “temp” data.

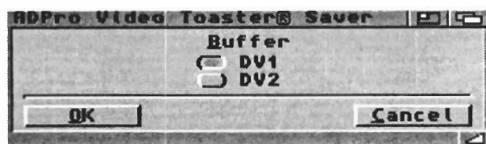


Figure 7.24: The VT_BUFFER saver control panel.

7.30 VT_BUFFER

The VT_BUFFER saver allows you to export the raw image data currently in ADPro's image buffer to one of the frame buffers (DV1 or DV2) of a currently running Video Toaster 2.0 (or later) system.

NOTE The Switcher needs to be running for this saver to work.

Select the buffer (**DV1** or **DV2**) to use from the control panel (shown in Figure 7.24).

Because the Video Toaster has a problem displaying images which are not exactly 752 pixels wide by 480 pixels tall, the saver will only save out images in this resolution. As such, if the image you are saving is smaller than this resolution, it will be composited centered on a black backdrop of this size. If your image is larger than this resolution, then the middle portion will be used.

If you want your entire image to appear in the frame buffer, you should either start off with the correct image size or scale it to the desired dimensions before invoking this saver.

7.31 _MultiCopyPREFPRINTER

This pseudo saver allows you to print multiple copies of the same image to your printer using the PREFPRINTER saver. The current PREFPRINTER settings are used, so set them up manually beforehand.

7.32 _SCULPTWithDimen

This pseudo saver allows you to save your raw image data in SCULPT format, but also includes the width and height in the filename so that you don't have to remember the dimensions of a SCULPT file when loading it.

7.33 _VT_Display

This pseudo saver allows you to save your raw image data to the Video Toaster's frame buffer. If the image is not 752 x 480, you will be asked whether it should be scaled to fill that area.

Although this function is duplicated by the VT_BUFFER saver, this pseudo saver is available if you want the entire image to be displayed—it will be scaled automatically. If you were to use the VT_BUFFER saver, images larger than 752 x 480 would be cropped, and not scaled.

NOTE Do NOT try to execute this pseudo saver if you don't have a Video Toaster and its software installed in your machine.

Chapter 8

Operators

This chapter provides detailed descriptions of the use and operation of all operators which are included with ADPro. ARexx control of these operators is described in Chapter 12.

8.1 Antique

The Antique operator applies an “old photograph” or antique, brownish look to an image.

Raw color image data is required for this operation. To apply to a grayscale image, first execute the `Gray_To_Color` operator.

Once the operator is executed, the meter window will appear showing the progress of the operation. After it has completed, the raw image data can be further modified by another operator or saved with one of the saver modules.

Note that if you had rendered image data before the operation, like most operators, you will have to rerender the data by pressing the **Execute** button.

8.2 Apply_Map

Changes made in the Balancing control panel, such as **Brightness:** or **Gamma:**, do not actually change any of the raw image data. Rather, these changes affect only look-up tables through which the raw image data is filtered. Therefore, the affect of changes made with these Balancing controls can

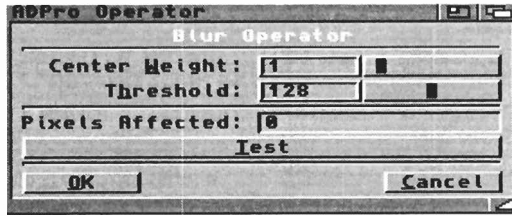


Figure 8.1: The Blur operator control panel.

be undone.

The Apply_Map operator applies the changes made with the Balancing controls to the actual raw image data. It then resets the Balancing controls to a neutral setting. The Apply_Map operator, once started, cannot be aborted.

Three uses of this operator come to mind:

1. It allows the range of the Balancing controls to be extended, by allowing you to repetitively apply color control changes over and over again.
2. It allows for better results when merging images using the image compositing feature of ADPro. For instance, you might want to merge an image with bright color settings with an image with dark color settings. Using this operator on both images prior to compositing will allow you to get the desired results.
3. It allows other Amiga programs which can read 24 bit-plane IFF images but do not parse "color look-up table chunks" to get the image data you intended.

8.3 Blur

Selecting the Blur operator causes the control panel shown in Figure 8.1 to be displayed. You can use this control panel to select or test the amount of blurring that will be done on an image.

The Blur operator allows you to blur certain pixels which differ from their neighbors by more than a user-definable threshold. Each pixel (weighted by a factor called the center weight) is averaged with the 8 surrounding pixels. If the resulting average differs from the actual pixel value by more than the user-defined threshold, the pixel is replaced by the average value. If the difference between the average and the original pixel does not exceed the threshold, the pixel is unchanged.

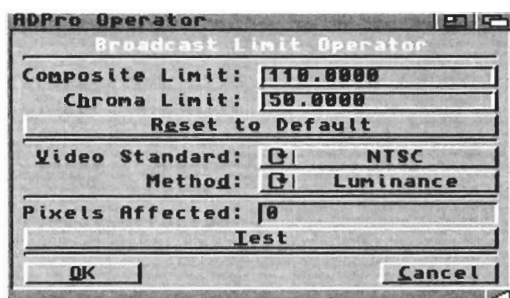


Figure 8.2: The Broadcast_Limit operator control panel.

A center weighting factor (**Center Weight:**) from 0 to 16 may be specified. A lower center weight causes a more dramatic blur of the image.

A threshold (**Threshold:**) of 0 to 255 may be specified. A value of zero forces all pixels to be blurred. A higher threshold will reduce the number of pixels which will be blurred.

The Blur operator allows a given set of values (for center weight and threshold) to be “pencil checked” using the **Test** button. By selecting this button, all computations will be made, but no changes will be made to the original data. The number of pixels which *would have been* affected will be reported. You can use this feature to fine tune your selections of center weight and threshold.

Selecting the **Cancel** button will exit from the operator without modifying the data. Selecting the **OK** button will apply the currently defined blurring operation to the raw data.

8.4 Broadcast_Limit

The Broadcast_Limit operator (shown in Figure 8.2) allows you to set the IRE limit of the composite video signal that would be generated if you were to display the currently loaded image on a NTSC or PAL display device. Video signal amplitudes are usually measured in IRE units, where 100 IRE is maximum white and 7.5 IRE is picture black (In PAL, picture black is the same as blanking black, which has an IRE value of 0).

When a picture reaches 120 IRE, video equipment may clip or distort the image being displayed. This may result in color errors in the area where the IRE limit exceeds 120. Colors, such as full-intensity yellow and cyan, encode to 131 IRE and, therefore, should be reduced before they are broadcast. An IRE value of 110 is a conservative limit for images which will be broadcast.

This will ensure the composite signal will not exceed 120 IRE even after being processed several times. Limiting the IRE to 100 is safer, but severely limits the range of colors in the image. A safe level for the chroma in an image is 50 IRE. If the chroma is higher, the image may be too saturated for an NTSC or PAL display.

The **Method:** cycle gadget will allow you to switch methods for correcting out-of-bounds colors. Setting this gadget to **Saturation** will reduce the amount of color in offending pixel. The **Luminance** setting will reduce the brightness in each offending pixel.

A **Composite Limit:** value of 0 to 199.9 may be specified. A value of 0 will cause all pixels to become black, probably not what you had in mind. Specifying a value greater than 135 will have little or no effect on a picture (but will also not give any broadcast benefits).

A **Chroma Limit:** value of 0 to 99.9 may be specified. When using changes in luminance to correct for broadcast, a value of 0 will cause all pixels to become black. However, when using saturation to correct for broadcast, a value of 0 will cause the color image to be converted into a grayscale image. Specifying a value greater than 60 will have little or no effect on the image.

The **Video Type:** cycle gadget allows you to switch between the default settings of the two major broadcast standards used today. The major difference between the **PAL** and **NTSC** settings are the value of picture black. In NTSC, picture black is 7.5 IRE, in PAL, picture black is 0.0 IRE.

The **Reset to Default** button resets the composite value to the 110.0 IRE standard, and the chroma value to the 50.0 IRE standard. The default values are the same for both NTSC and PAL machines.

The Broadcast_Limit operator allows a given set of values to be “pencil checked” using the **Test** button. By selecting the **Test** button, the computations will be performed but no changes will be made to the original data. The number of pixels which would have been affected will be reported in the **Pixels Affected:** text field.

Selecting the **Cancel** button will exit from the operator without modifying the data. Selecting the **OK** button will apply the currently composite and chroma values to the raw data.

8.5 Collapse

The Collapse operator (shown in Figure 8.3) allows you to take a circular portion of your image (or the entire image itself) and “pull” it toward the center of the circle. This operator can be used to create caricatures of a

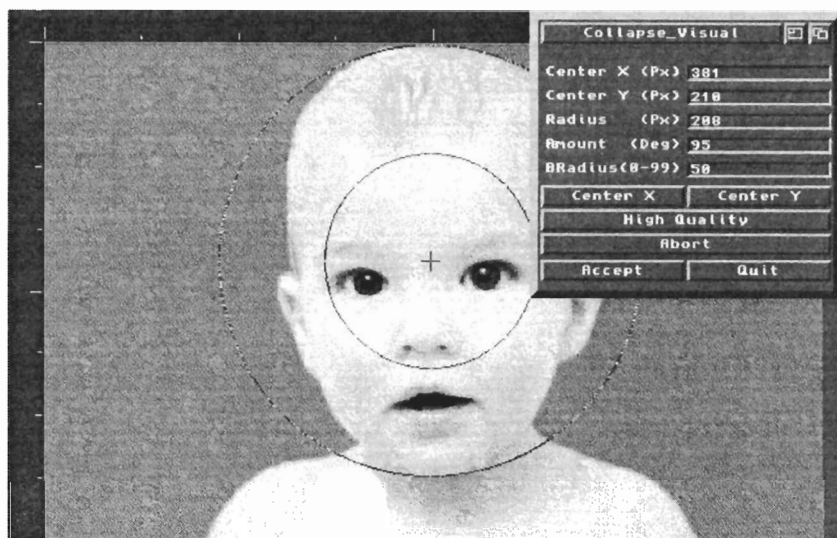


Figure 8.3: The Collapse operator control panel.

person's face or other artistic effect.

When the grayscale representation of your image is first being drawn on the screen, you can interrupt it by pressing the **Abort** button.

The center of the collapse circle (the area which will be collapsed) can be edited by dragging the circle's crosshair around on the screen or typing values into the **Center X (Px)** and **Center Y (Px)** input fields. You can center the circle horizontally, vertically, or both on the image by using the **Center X** and **Center Y** buttons.

The size of the collapse circle is described by its radius and defined in the **Radius (Px)** input field. You can also click virtually anywhere on the screen to adjust the circle's size. The circle will expand to contract to the location of the mouse pointer.

The amount of collapsing can be controlled by the value in the **Amount (Deg)** input field. The higher the amount the greater the effect, and vice versa.

The area near the outside edge of the circle can be "blurred" by specifying a value in the **BRadius(0-99)** input field. This value represents a percentage of the radius (starting from the outermost part of the circle going inward) to be blurred. A value of 0 means no blurring, while a value of 99 means that the entire circle will be blurred.

The quality of the effect can be controlled by toggling the quality rocker

between **Fast** and **High Quality**. **Fast** works faster, but may not produce as precise an image as **High Quality**.

Press the **Accept** button to perform the collapse operation, or **Quit** to return to the main window without performing the operation.

8.6 Color_To_Gray

The Color_To_Gray operator provides three very important benefits beyond simply turning color images into grayscale images:

1. The method used within ADPro to convert from color into grayscale can actually produce grayscale data of considerably better quality than the original color data. For example, the Sharp JX-100 portable scanner is accurate in grayscale to 6 bit-planes and 18 bit-planes in color. However, when ADPro converts an 18 bit-plane color image into grayscale, it produces a grayscale image which is accurate to 8 bit-planes.
2. ADPro renders extremely well in grayscale due to its dithering capabilities and the fact that 16 grayscales cover the range of gray proportionately better than 16 colors cover the spectrum of all possible colors. For this reason, a high resolution interlaced gray image rendered with ADPro can look very nearly photographic, even given the Amiga's limited display capability.
3. This feature provides a way to convert color images into high quality grayscale images for the purpose of black and white desktop publishing. While color DTP is just now coming into its own, DTP is still predominantly grayscale.

Selecting the Color_To_Gray operator causes the control panel shown in Figure 8.4 to be displayed. You can use this control panel to select the weights with which the red, green, and blue components of the image will be factored into the color to grayscale conversion.

Two pre-programmed weighting schemes are provided. These can be enabled by selecting either the **Average Weights** or **Luminance Weights** button. By selecting the button marked **Average Weights**, each of the red, green, and blue components of the image will be given equal weight as the image is converted to grayscale. In essence, each gray pixel will be the average of the red, green, and blue color pixel components.

Selecting the button marked **Luminance Weights** will set up the weighting scheme used by the NTSC television standard to convert a color broadcast signal for display on a black-and-white television receiver.

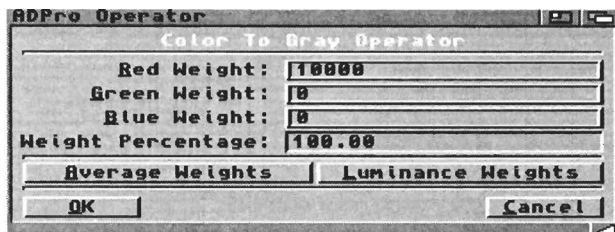


Figure 8.4: The Color.To.Gray operator control panel.

If you wish to specify your own weights, you may do so by entering values into the three integer input fields supplied. The values you enter represent the fractional contribution that each color will have in the final averaging process. These values are scaled such that 0 represents 0 percent weight while 10000 represents a 100 percent weight. The total weighting is given below the three integer input fields and may exceed 100 percent.

In Figure 8.4, a weight of 10000 has been given to the red component while a weighting of zero has been given to the other colors. This will cause pixels with a large red component in the original to be lightly shaded in the resulting gray scale image. Pixels with very little red will be shaded darkly in the resulting gray scale image.

A total weighting percentage may exceed 100 percent. Pixels in the resulting grayscale image are clipped to the range of 0 to 255. Negative weights may also be specified.

Selecting the button marked **OK** causes the currently defined conversion to take place. Selecting the button marked **Cancel** exits the operator without changing the current image in memory.

8.7 Colorize

This operator is an alternative to Color.To.Gray in that both operators will reformat grayscale image data into color image data within ADPro's memory. However, Color.To.Gray simply creates grayscale data in the format of color data, whereas Colorize will actually color in the grayscale image.

Recall that grayscale image data is stored in 256 shades within ADPro.

The Colorize control panel is shown in Figure 8.5. Setting the **Type:** to **Random**, the 256 shades of gray will be converted into 256 colors chosen at random. The color choosing is not completely random as this would produce an unrecognizable image. Rather, the random color choices will center around

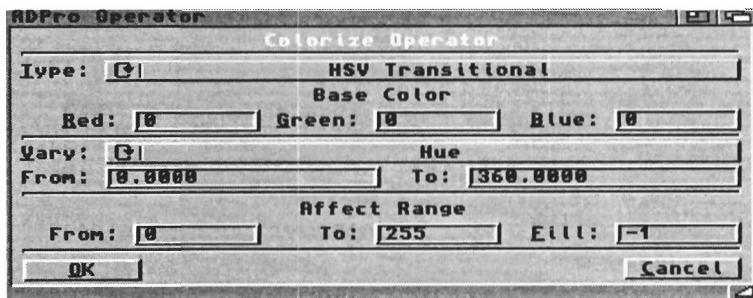


Figure 8.5: The Colorize operator control panel.

the color you specify in the requester (in the **Base Color** input fields).

You can convert grayscale into color using the currently loaded color palette if you select the **Currently Loaded** setting. You can load a palette in from disk prior to executing this operator if you wish to match a specific palette. Remember that in a grayscale image, BLACK is generally color register 0 and that WHITE is generally color register 255.

You can produce a false colorization with the **False Color** setting. A false colorization puts cold blues in the darker range, and hot reds in the brighter areas.

You can produce a wide range of effects with the **HSV Transitional** setting. These will be explained later.

You can specify a base color using the integer input fields provided. The **Base Color** is used by the **HSV Transitional** and **Random** settings.

NOTE The base color is used to set parameters for the **HSV Transitional** colorization, which means that setting the **Base Color** to specific values like BLACK will cause an entirely BLACK colorization (probably NOT what you had in mind).

When the **Vary:** cycle gadget is set to **Hue**, the value and saturation of the colors used in the colorization will be taken from the **Base Color**. The spread of hues created can range between 0 and 10000. Note that hues range from 0 to 360. Specifying a value larger than 360 will result in a cyclic repetition of colors.

When **Vary:** is set to **Saturation**, the hue and value of the colors used in the colorization will be taken from the **Base Color**. The spread of saturation which will be used can range from 0 to 1.

When **Vary:** is set to **Value**, the hue and saturation of the colors used in the

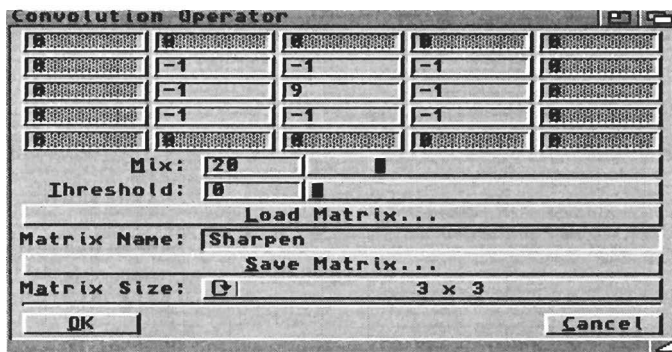


Figure 8.6: The Convolve operator control panel.

colorization will be taken from the base color. The spread of value which will be used can range from 0 to 1.

With all types of colorization, you can select a range of gray shades which will be affected. The range always flows from the number on the left, upwards to the number on the right. This means that you can specify a range which wraps around the 255 (white) mark back to the 0 (black) mark. Shades of gray which are outside the range can either be left alone (a **Fill**: value of -1) or set to the specific intensity of gray (0 to 255), which is typed into the **Fill**: input field.

For example, you might specify a false colorization between shades 0 and 127 with the fill value set to -1. This means any pixel which had a gray shade between 0 and 127 (inclusive) will be false colored in the new color image. Pixels from 128 to 255 will be the same shade of gray they were originally.

If you wished to colorize gray shades starting at 250 and wrapping around to 20, then you would specify shades 250 and 20 in the **From**: and **To**: input fields, respectively.

8.8 Convolve

The Convolve operator is a general purpose convolution filter which can apply 3x3 or 5x5 convolution matrices to the image data. Convolutions can perform many different image processing functions by varying the values comprising the convolution matrix. The control panel for this operator is shown in Figure 8.6.

As configured in Figure 8.6, the Convolve operator is ready to perform a sharpening function. Determining the values to use in the convolution matrix requires either much experimentation or a knowledge of image processing theory.

To save you from either of these, we provide a large number of well-known convolution matrices with the ADPro distribution.

To load a convolution matrix, use the button marked **Load Matrix...** Selecting it will cause the file requester to appear. Use the file requester to select the convolution of your choice. The filename of the currently loaded matrix is displayed in the **Matrix Name:** field. Similarly, a given convolution matrix can be saved using the button marked **Save Matrix...**

The **Matrix Size:** cycle gadget can be set to a **3 x 3** or **5 x 5** convolution matrix.

The **Mix:** and **Threshold:** values interact as follows: After a given pixel has been convolved, the new value for that pixel is compared with the old value for that pixel. If the difference between these two values does not exceed the threshold, the new value is thrown away and the old value is retained. If the new value is to be used (i.e., the difference between the old and new values for the pixel exceeds the threshold), then the new value is mixed into the old value based upon the mix setting. A low mix setting means that the new value will affect the old value only slightly. A large mix setting means the new value will greatly affect the old value, perhaps even replacing it completely.

You will find that convolutions affect the image data very severely. Therefore, you will probably want to use a low value for the mix setting so as to mute the effect of the convolution. The setting shown in Figure 8.6, for example, is a reasonable value for the **Sharpen** matrix. Figure 8.7 shows a ray tracing by Allen Hastings called "Hallway." The left half of the image has been sharpened using the Convolve operator, the right half has not.

As you would expect, 5 x 5 convolutions take longer to execute than 3 x 3 convolutions.

8.9 Crop_Image

The **Crop_Image** operator can be used to specify a rectangular region within an image which will be preserved. All data outside the selected region will be discarded.

NOTE Usage of this operator from the user interface is no longer suggested. Instead, use the **Crop_Visual** operator which provides you with an on-screen representation of the crop operation, rather than the numeric-only user interface which this operator presents. This operator is retained for use through **ARexx** control where the visual capabilities of **Crop_Visual** will not be used. See Section 8.10 for more information about the **Crop_Visual** operator

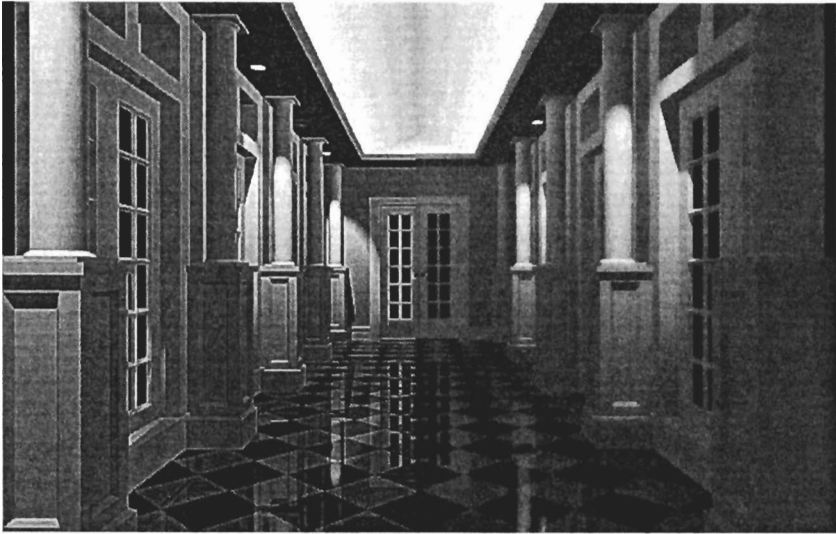


Figure 8.7: The left half of this ray tracing has been sharpened using the Convolve operator.

When the Crop_Image operator is selected, it will bring up a control panel (shown in Figure 8.8) into which you can specify the offset (**X Offset:** and **Y Offset:**) of the top left-hand corner of the rectangle to be preserved. You specify the **New Width:** and **New Height:** of the area to be preserved as well. Note that you cannot specify a negative offset nor can you specify a combination of offsets and width or height which would exceed the width or height of the original image.

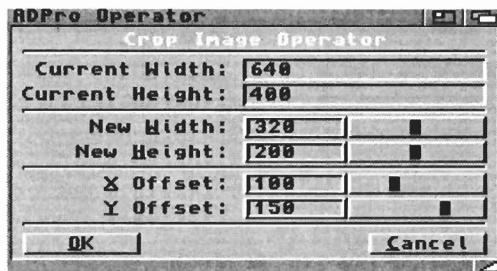


Figure 8.8: The Crop_Image operator control panel.

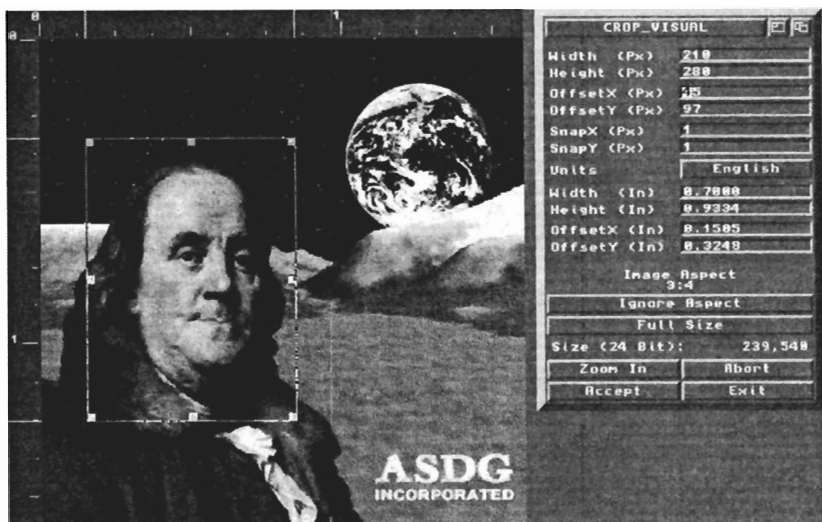


Figure 8.9: The Crop_Visual operator control screen.

Aside from the straightforward use of this operator for cropping, you can also use this operator to “focus” ADPro’s color picking technology. See **Focused Color Picking** in Appendix B for more information.

8.10 Crop_Visual

This operator provides an on-screen WYSIWYG user interface to the cropping operation. The **Crop_Visual** control panel is shown in Figure 8.9.

The box representing the area of interest indicates where the cropping operation will take place. The portion of the image inside the box will be retained, while the area outside the box will be discarded. You can use the buttons marked **Full Size** and **Zoom In** to see finer or grosser detail. The **Accept** button will cause the cropping operation to actually take place. The **Cancel** button will leave the Crop_Visual operator without performing a crop.

Pressing **Shift + Return** on your keyboard is the same as hitting the **Accept** button. When the control panel is zipped, hitting the **z** key on your keyboard is the same as hitting the **Zoom In** button. Also, when the panel is zipped, hitting the **u** key on your keyboard will unzip the panel, making it full size again.

8.11 DCTV

The DCTV operator reformats the raw image data currently loaded in ADPro's memory into the format used by the DCTV display device (from Digital Creations). The reformatting overwrites the rendered image area in ADPro's memory but does not disturb the raw image area. If ADPro is not already set for 8 colors, the operator automatically shifts ADPro into Hi Res, 16 colors as required for DCTV. After the operator completes, it will display the resulting DCTV formatted image. This image cannot be scrolled from within ADPro.

If ADPro was already set to 8 colors, a 3 bit-plane DCTV image will be created instead of a 4 bit-plane version.

NOTE Note that if your current display module is a 24-bit graphics board, it will be switched to an NTSC:Hires 4 bit-plane mode. It will only automatically switch into interlace mode if the previous mode was interlaced.

The DCTV operator takes its display size and type (i.e., interlace on or not, PAL or NTSC, etc.) from ADPro's current screen mode setting.

You can save the DCTV version of the image by selecting the IFF saver. When saving a DCTV image, make sure you select the button marked *n* **Color Image** where *n* is either 8 or 16.

You can use the IFF loader to load a DCTV image into ADPro.

To decode a DCTV encoded image into 24-bit color data, read Section 8.33.

NOTE This operator requires and uses the `dctv.library` available from Digital Creations. If this library is not present in your **LIBS:** directory, you will get a message indicating a "fatal error in the operator". This library is included with this version of ADPro.

8.12 Define_Pxl_Aspect

ADPro adjusts the pixel aspect of the image as the image is scaled. After many scaling functions, the pixel aspect of the image may have little information left in it. For example, after scaling an image several times, you might arrive at an image that you wish to use as a low resolution interlaced picture. You don't care what the pixel aspect was before you started, but now that you have a picture you want to keep, you might want to redefine the pixel aspect to be exactly right for low resolution interlaced.

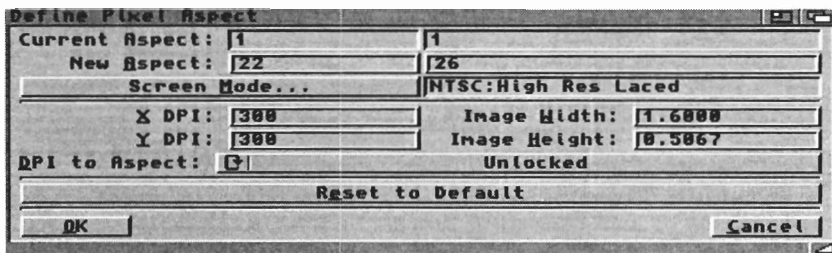


Figure 8.10: The Define_Pxl_Aspect operator control panel.

Also, many applications (such as desktop publishing) can make use of other resolution and sizing information which some file formats can maintain.

Using the Define.Pxl.Aspect operator, you can redefine the pixel aspect of the currently loaded image to be whatever value you wish. You can also redefine the “resolution” of the image so that programs such as desktop publishing packages will believe your image should be whatever size you want it to be.

Figure 8.10 shows the control panel for this operator. It displays the current pixel aspect (**Current Aspect:**) and has two integer input fields (**New Aspect:**) which allow you to enter a new pixel aspect numerically. You can also specify a new pixel aspect by selecting the **Screen Mode...** button and choosing a screen from the list of available modes.

Further, you can define an x and y resolution for your image. These values are maintained by ADPro and are actually used (in creating the rulers) in the operators.

NOTE This operator does not actually modify any of the image data. It simply redefines ADPro’s internal idea of the image’s pixel aspect and resolution. The new pixel aspect and resolution information will be saved with the image when the image is saved in a format which supports such information.

8.13 DeInterlace

The DeInterlace operator separates the currently loaded image into two images, stacked one upon the other. The top image is comprised of all of the even numbered scanlines (lines 0, 2, 4, etc.) from the original image. The bottom image is comprised of what were the odd numbered scanlines of the original image.

This operator is extremely useful for processing images grabbed with video

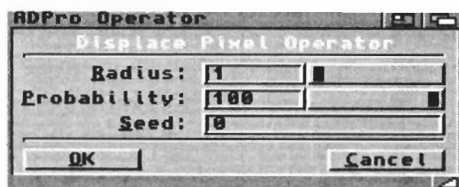


Figure 8.11: The Displace_Pixel operator control panel.

digitizers such as the GVP IV24 and the Video Toaster (the Progressive Peripherals, Inc. FrameGrabber supports deinterlacing in hardware).

Such devices grab a full video frame which is comprised of two video fields, one sixtieth of a second apart (or one fiftieth of a second apart in non-NTSC regions). The time lag between fields can produce unwanted results particularly if the image being digitized contains rapidly moving elements. The DeInterlace operator can split the image data from the two fields apart so that either field may be processed separately.

To crop out either the top or bottom half of the image (i.e., the even or odd field), see the discussion on the Crop_Visual operator.

Note that the DeInterlace operator affects ADPro's impression of pixel aspect since the deinterlaced image would be properly shown on a screen half as high as the original image.

8.14 Displace_Pixel

The Displace_Pixel operator (shown in Figure 8.11) allows you scatter the pixels in an image. This, along with image compositing, can be used to create dramatic dissolves.

The **Radius:** value defines the circular area in which any given pixel can be displaced from its current position.

The **Probability:** value defines the likelihood that a given pixel will be displaced. This value can range between 1 and 100, where lower values mean that the pixel will less likely be to be displaced and higher values mean a greater chance.

The **Seed:** value "re-seeds" the randomizer, which determines in what direction a given pixel will be displaced. If you leave this value constant between invocations, you will get the same results. Changing it allows you to produce different results.

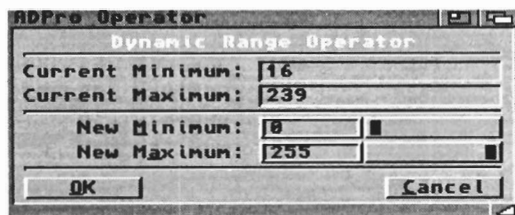


Figure 8.12: The Dynamic_Range operator control panel.

Press the **OK** button to execute this operator on the image. Press the **Cancel** button to exit this operator without performing the operation.

8.15 Dynamic_Range

The Dynamic_Range operator performs two passes over any raw image data currently loaded in memory. During the first pass, the smallest and largest values contained in the raw image are found. Then, a control panel (shown in Figure 8.12) appears, displaying the values that were found in the first pass. If you select and accept a **New Maximum:** and **New Minimum:** value, a second pass over the raw image is performed, mapping all of the contained image data to lie between the selected maximum and minimum.

This operator can be used in many different ways. It can be used to expand or contract the dynamic range of an image (by increasing or decreasing the spread between the maximum and minimum values). It can also be used as a hybrid contrast/brightness manipulation.

Since this operator allows colors to be uniformly compacted, video users can use this operator to prevent NTSC smear. This can be done by entering a new maximum value of approximately 208 rather than 255. Doing this causes the most intense color component picked for an Amiga palette to be 13 or less.

8.16 Gray_To_Color

The Gray_To_Color operator provides the logical inverse of the Color_To_Gray operator. Gray_To_Color reformats the internal data representation of raw grayscale image data into that of raw color image data. This feature is provided so that you can perform all image processing functions on both color and grayscale images.

When combined with image compositing, interesting artistic effects can be

created. For example, a masked color image can be “pasted” into an otherwise completely grayscale image for a startling effect. See **Merging a Color Image Into a Grayscale Image** in Appendix B for more information.

8.17 Halve

The Halve operator is a highly optimized version of ADPro’s scaling function designed to halve the width and height of the currently loaded image. This operator retains all of the precision of ADPro’s scaling function, but executes approximately 13 times faster (on a 68030 processor). This operator is especially useful for CDTV/CD32 developers who might frame grab thousands of images at video resolution but needs the results in quarter-screen size.

8.18 Horizontal_Flip

Many applications require the production of mirror images of the original data. Specifically, this is required by screen and heat-transfer printers. Selecting the Horizontal_Flip operator will horizontally flip the raw image data currently in memory.

Note that performing a horizontal flip followed by a vertical flip is the same as rotating the image 180 degrees. This can be used in conjunction with the Orientation controls (**No Composition**, which will do a Portrait load, and **Landscape**) to rotate images through 0, 90, 180, and 270 degrees. Note, as well, that you can also do these same rotations with the Rotate operator.

8.19 Hist_Equalization

The Hist_Equalization (Histogram Equalization) operator allows you to “even out” (equalize) the luminance, or brightness, of an image. It is a two pass operation, in which the first pass takes statistics on the luminance levels in the image and the second pass modifies these levels so that they will be spread smoothly across the 0 to 255 luminance range.

In simpler terms, darker images become lighter, lighter images become darker, and images with fairly equal amounts of light and dark don’t change much.

Raw color or grayscale image data is required for this operation.

Once the operator is executed, the first meter window will appear showing the progress of the histogram building phase. After this, the meter will be

restarted for the equalization phase. After these two phases, then the raw image data will be “equalized”.

Note that if you had rendered image data before the operation, like most operators, you will have to rerender the data by pressing the **Execute** button.

If you find that the results are lighter or darker than what you expected, then compositing the original image onto this modified image at various mix levels (see Section 5.3.3) should give you a better image.

8.20 Intensity_Range

NOTE The Intensity_Range operator was developed by Steve Tibbett, and not ASDG, Incorporated. As such, you should contact Steve Tibbett’s BBS at (613) 731-3419 for technical support. This module is included in the ADPro package under agreement with Steve Tibbett.

The Intensity_Range operator will build a graph showing you the relative number of pixels of varying intensities in the image, and let you bring the ends in, cutting out either the top, bottom, or both ends of the intensity range.

To use it, load or create a 24 bit image (gray and rendered images are not supported, but you can convert either of these into 24 bit with other ADPro operators), select the Intensity_Range operator and operate. Intensity_Range will build a histogram to show you 256 intensity levels (across the histogram) and the amount of the image at each intensity level. The graph is scaled so that the highest concentration of intensity in the image is the height of the graph. The marks below the histogram are initially set to the top and bottom end of the intensity scale—when you move these and select **Accept**, the image will be remapped so that the area between the two marks will fill the entire intensity range. Any pixels whose intensity is above where you set the top mark are set to white, and pixels below where you set the bottom mark are set to black.

An example of the usage of this is taking a scanned image and running this operator on it: If the histogram shows that most of the image is grouped around one area, and that there are very few pixels outside that area, you might try bringing the ends in so that they encompass the bulk of the image. Cutting off an unused bit of the high or low end can sometimes improve an image quite a bit.

If an image is too dark, for example, bringing up the brightness will increase the brightness of the whole image—leaving you with no black in the image at all. Bringing the top end down to where the bulk of the image ends, will do a

much better job of making a dark image seem brighter.

An interesting side effect is that you can tell what color resolution the image was created with. The histogram shows 256 intensities, but any standard Amiga image only has 4 significant bits—so you will only see a maximum of 16 spikes in the histogram. A scanned 24 bit image will usually show a smooth curve, and that is the kind of image that this operator works best with.

If you want to look for detail in the dark area of an image, then drag the top end down to near the bottom end and press **Accept**. Same for any other part of the image intensity range.

This operator works almost exactly like the `Dynamic_Range` operator that comes with ADPro, except that with `Dynamic_Range` you only know the highest and lowest intensities in the image, but not how the image really looks—one pixel at 255 and one at 0 will fool the `Dynamic_Range` operator.

This operator only works on 24bit image data.

8.21 Interlace

The Interlace operator facilitates the creation of interlaced images whose odd and even fields are composed of different pictures. This operator is the logical inverse of `DeInterlace`.

To interlace two different images together, use the `BACKDROP` loader to create a workspace tall enough to accommodate both images (i.e., twice as tall as either image alone). Then, using the compositing capability, load one image into the top half of the backdrop, and the other image into the bottom half of the backdrop. Then execute this operator.

The resulting image will contain alternate scanlines from both of your original images.

Note that the Interlace operator affects ADPro's impression of pixel aspect since the interlaced image would be properly shown on a screen twice as high as the original images.

8.22 KillTemp

The `KillTemp` operator allows you to clear the image currently in ADPro's temporary buffer (placed there by the `TEMP` saver), whether or not an image exists in the buffer.

To determine if image data is currently in the temporary buffer, select the **About...** menu item under the **Project** menu to display the information about the program. If an asterisk (*) appears to the right of the **Buffer Size:** value, then the temporary buffer is not empty.

8.23 Line_Art

When 8 bit-plane grayscale data is available, selecting the **Line_Art** operator invokes a proprietary edge emphasis algorithm. The result is an image which contains the most prominent edges in the original image.

The production of quality line art from an arbitrary grayscale bit-map is quite difficult. It is important to begin with a high contrast original or to boost the contrast of the image before invoking the **Line_Art** operator.

Modifying the color controls (such as Brightness, Contrast, and Gamma correction) prior to executing the **Line_Art** operator will drastically affect the quality of the resulting line art. Increasing the brightness, contrast, or gamma correction of the image will also reduce the amount of clutter in the resulting line art. In general, increasing the contrast setting is usually a good step.

After the **Line_Art** operator has been executed, the color controls, such as brightness, contrast, and gamma correction, will only slightly affect the rendered image. Decreasing brightness or contrast will thicken the lines in the image. Increasing brightness, contrast, or gamma correction will thin out the lines in the rendered image.

When working with the **Line_Art** operator, we suggest that you save the original 8 bit-plane grayscale data to disk before executing the operator. In this way, you can reload the original 8 bit-plane grayscale data and retry the command with different brightness, contrast, and gamma correction settings.

NOTE Executing the **Line_Art** operator will reduce the size of the available image data by 2 pixels in width and height.

8.24 Median_Filter

The **Median_Filter** operator computes the median color for each pixel in the image based upon itself and the 8 surrounding pixels. If the center pixel differs from the computed median by more than a specified threshold, the pixel is replaced with the computed median.

This operator can reduce noise or small unwanted artifacts in an image by

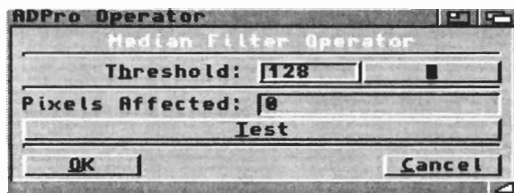


Figure 8.13: The Median Filter operator control panel.

replacing pixels which “seem not to fit in” with values which match the surrounding pixels more closely.

You specify a **Threshold:** which determines how greatly a given pixel must differ from its surrounding pixels before it will be replaced. A lower threshold (which may range from 0 to 255) causes more pixels to be affected.

NOTE Each of the primary colors is considered separately. This can sometimes introduce artifacts rather than obscure them.

The Median Filter operator allows a given threshold to be “pencil checked” using the **Test** button. By selecting this button, all computations will be made, but no changes will be made to the original data. The number of pixels which *would have been* affected will be reported in the **Pixels Affected:** text field. You can use this feature to fine tune your selection of **Threshold:**.

Selecting the **Cancel** button will exit from the operator without modifying the data. Selecting the **OK** button will apply the currently defined median filter operation to the raw data.

8.25 Mosaic

The Mosaic operator (shown in Figure 8.14) allows you to create a “tiled” representation of your image, similar to the effect used for murals.

The **Width:** and **Height:** values specify the dimensions of each “tile”, while the **X Offset:** and **Y Offset:** values define the upper-left location of the first full “tile” relative to the image’s borders.

Press the **OK** button to create a mosaic using the defined settings, or **Cancel** to return to the main window without performing the operation.

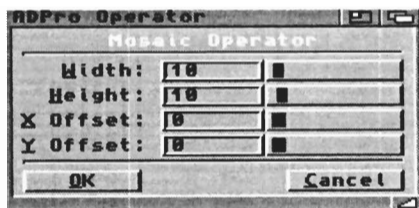


Figure 8.14: The Mosaic operator control panel.

8.26 Negative

The Negative operator inverts any raw image data currently loaded in memory. For grayscale data, this produces what would commonly be called a black-and-white negative image of the original data. For color image data, each primary color is inverted separately.

This operator requires the presence of raw image data in memory.

8.27 OpalPaint

NOTE The OpalPaint operator was developed by Opal Technology, and not ASDG, Incorporated. As such, you should contact Opal Technology for technical support. This module is included in the ADPro package under agreement with Opal Technology.

Also, you must have at least version 1.1 of the OpalPaint program (not the ADPro operator) and enough memory to run the OpalPaint program (you might need to use the **MAXMEM** Tool Type). The latest versions of the OPALVISION software is available directly from Opal Technology / Centaur Development or their support BBS at (310) 793-7142.

Be sure you have the proper **Assigns** applied before executing this operator.

The OpalPaint operator allows you to use the OpalPaint program (included with the OPALVISION display board) to touch-up or modify the raw image data in ADPro.

After selecting this operator and a short delay you will be presented with the OpalPaint program's screen, but with your raw ADPro image already loaded into the program. You can then use all the tools available in the OpalPaint program to edit the image.

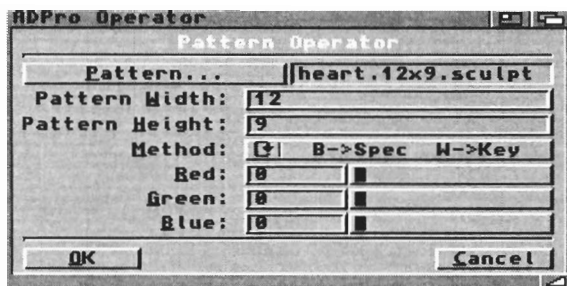


Figure 8.15: The Pattern operator control panel.

Once you exit out of OpalPaint, and if you have modified the image, a requester will appear asking you if the changes should be preserved. If you accept these changes, the modified image will be reloaded into ADPro. Otherwise, your original image will be restored.

If you are running OpalPaint version 2.3 or later, the OpalPaint program will also ask if you want to keep it resident in memory. If you choose this option, then the next time you execute the OpalPaint operator, it won't take as long to come up.

8.28 Pattern

The Pattern operator is one of the most interesting and flexible operators available in ADPro. By using SCULPT files as the “pattern” which will be repeated, you can create some interesting images.

The Pattern control panel is shown in Figure 8.15. To use an existing pattern (SCULPT) file, select the **Pattern...** button and choose the pattern to use. The pattern will be repeated in a grid pattern atop the image.

Enter the width and height of the pattern in the **Pattern Width:** and **Pattern Height:** input fields.

NOTE Since SCULPT files do not contain the size information within the file, you must remember what they are. A good tip is to embed the size in the filename, as in “TestPattern_10x10.sculpt”. All the bundled pattern files use this tip, so you'll know what to enter into the input fields.

The **Method:** cycle gadget controls how the pattern is used. **B** represents the black parts of the the pattern, whereas **W** represents the white parts. You can

control how the black and white parts of the pattern manipulate the image. Use the following as a guide:

Spec	This option replaces the part with the color specified by the Red: , Green: and Blue: values.
Orig	This option replaces the part with the original image underneath that part, letting you effectively produce a transparent area.
Key	This option replaces the white areas of the pattern with 100% of the color located at the center of the pattern, with gray areas being replaced with a percentage of the color, and black being replaced with 0%.

If either black (**B**) or white (**W**) is set to **Spec**, the **Red:**, **Green:**, and **Blue:** input fields and sliders will become active for the specification of a color.

Select the **OK** button to apply the pattern to the image, or **Cancel** to return to the main window without changing the image.

It's very easy to create your own patterns. Simply use a paint program to create a grayscale brush with an interesting pattern. Save it out, load it into ADPro, and save it out in SCULPT format for use with this operator.

8.29 Polar_Mosaic

The Polar_Mosaic operator (shown in Figure 8.16) allows you to create a “tiled” image similar to the Mosaic operator, but using concentric circles and “pie-style” cuts. Think of the mosaic circle as a pie that you can make radial (from the center point outward) slices, as well as concentric slices (tracks).

When the grayscale representation of your image is first being drawn on the screen, you can interrupt it by pressing the **Abort** button.

The center of the mosaic circle can be modified by either dragging the on-screen circle around the screen or typing the circle's center coordinates in the **Center X (Px)** and **Center Y (Px)** input fields. Notice that you get immediate feedback of your changes. You can center the circle horizontally, vertically, or both on the image by using the **Center X** and **Center Y** buttons.

The size of the mosaic circle is described by its radius and defined in the **Radius (Px)** input field. You can also click virtually anywhere on the screen to adjust the circle's size. The circle will expand to contract to the location of the mouse pointer.

The number of “pie slices” to be made in this circular area is defined by the value in the **Slices** input field.

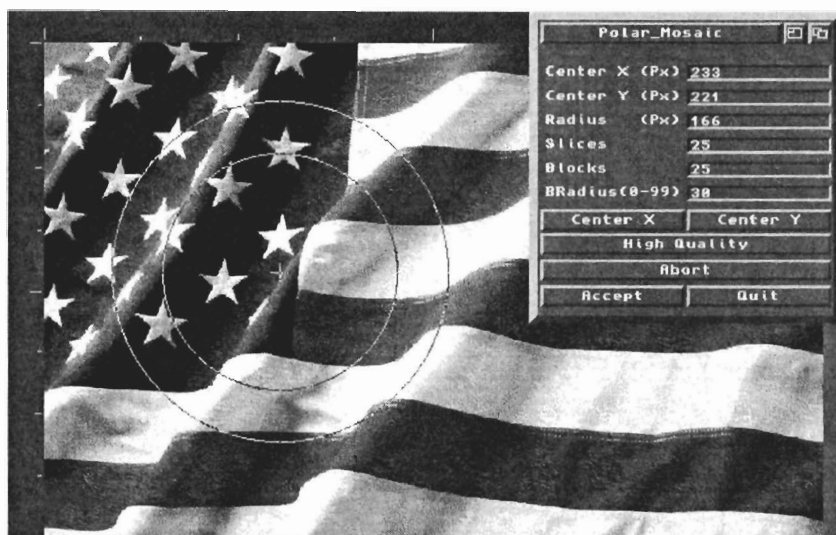


Figure 8.16: The Polar_Mosaic operator control panel.

The number of concentric “tracks” is defined in the **Tracks** input field. The more tracks you create, the more the original image will be recognizable in the modified image, and vice versa.

The area near the outside edge of the circle can be “blurred” by specifying a value in the **BRadius(0-99)** input field. This value represents a percentage of the radius (starting from the outermost part of the circle going inward) to be blurred. A value of 0 means no blurring, while a value of 99 means that the entire circle will be blurred.

The quality of the effect can be controlled by toggling the quality rocker switch between **Fast** and **High Quality**. **Fast** works faster, but may not produce as precise an image as **High Quality**.

Press the **Accept** button to perform the operation, or **Quit** to return to the main window without performing the operation.

8.30 Rectangle

The Rectangle operator can be used to draw filled and unfilled rectangles into a color or grayscale image, such as for borders around or drop shadows for rectangular regions. Selecting this operator causes the Rectangle control panel to appear. This control panel is shown in Figure 8.17.

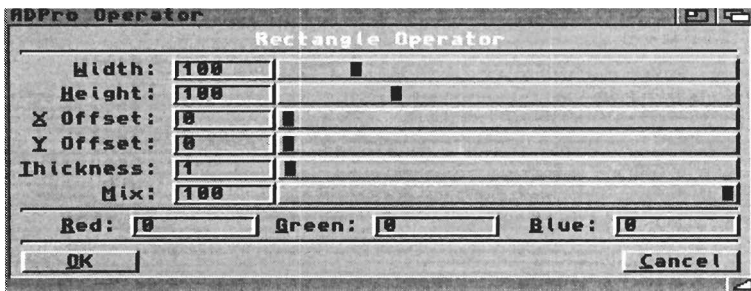


Figure 8.17: The Rectangle operator control panel.

NOTE Use of this operator from the user interface is no longer suggested. Instead, use the `Rectangle_Visual` operator which provides you with an on-screen representation of the operation, rather than the numeric-only user interface which this operator presents. This operator is retained for use through ARexx control where the visual capabilities of `Rectangle_Visual` will not be used. See Section 8.31 for more information about the `Rectangle_Visual` operator.

Using this control panel, you can specify the location of the upper left-hand corner (in the **X Offset:** and **Y Offset:** input fields) of a rectangle as well as its **Width:** and **Height:**. You can also specify the thickness (**Thickness:**) of the rectangle. The default thickness is 1, which means that a 1 pixel thick rectangle will be drawn. Specifying a larger value means that a thicker rectangle will be drawn. Since you specify the *outside* dimensions of the rectangle, any additional thickness will be drawn within the inside of specified region. A thickness of -1 causes a filled rectangle to be drawn.

You may also specify a **Mix:** value to be used in the drawing computation. A mix value of 100 is the default, which indicates that drawn pixels shall completely obscure any pixels from the original image which they overlap.

Lastly, you may specify the **Red:**, **Green:**, and **Blue:** content of the color to be used in drawing the rectangle. When drawing a rectangle in a grayscale bit map, only the red value is used to determine the shade of the rectangle. Colors are specified in the range of 0 to 255.

8.31 `Rectangle_Visual`

This operator provides an on-screen WYSIWYG user interface to the rectangle operation. We suggest that you use this operator rather than the `Rectangle` operator when working with ADPro from its user interface. However, when

working from ARexx, we suggest that you continue to use the Rectangle operator since its small size will allow it to load from disk more quickly than Rectangle_Visual. Also, from ARexx, the visual benefits of Rectangle_Visual are negated.

The Rectangle_Visual control panel is shown in Figures 8.18 and 8.19. Figure 8.18 shows how a rectangle which is to be completely filled is represented on-screen. When the rectangle is drawn with an “X” through it, it means that the entire interior of the rectangle will be filled. Figure 8.19 shows a rectangle which will not be completely filled. Instead, it will be drawn with a thickness of 20. In all cases, the handles on the outer rectangle are what you use to control the size and location of the rectangle.

You can set the thickness of the rectangle using the input field marked **Thickness**. If set to -1, the rectangle will be filled, regardless of its size. If set to a number larger than -1, the rectangle will be drawn with the appropriate thickness. Note that having a thickness means that, at some sizes, the box will be completely filled. Suppose you defined a thickness of 50. Clearly, if the box were less than 100 pixels high or wide (50 pixels on both sides of the rectangle), the box would be completely filled. However, if the box were 500 pixels wide and tall, it would not be completely filled.

You can specify how much pixels being drawn as part of a rectangle will replace the background using the input field marked **Mix**. You can specify the color of the rectangle to be drawn using the input fields marked **R**, **G**, and **B**.

Selecting the **Accept** button will cause the currently specified rectangle to be drawn. When the drawing is complete, the screen will be updated and you can specify another rectangle. To exit the operator, press the **Cancel** button.

Pressing **Shift + Return** on your keyboard is the same as hitting the **Accept** button. When the control panel is zipped (in its “iconified” form), pressing the **z** key on your keyboard is the same as selecting the **Zoom In** button. Also, when the panel is zipped, pressing the **u** key will unzip the panel (make it full size again).

8.32 Rem_Isolated_Pxls

The Rem_Isolated_Pxls (RIP or Remove Isolated Pixels) operator is used to “clean up” an already rendered image. It does so by examining each pixel in the rendered image data in succession. If the surrounding 8 pixels are the same color and the center pixel is a different color, the center pixel is replaced with the surrounding color.

We have found that Rem_Isolated_Pxls can “clean up” a majority of unwanted artifacts when rendering images with a small number of colors, such as a colored

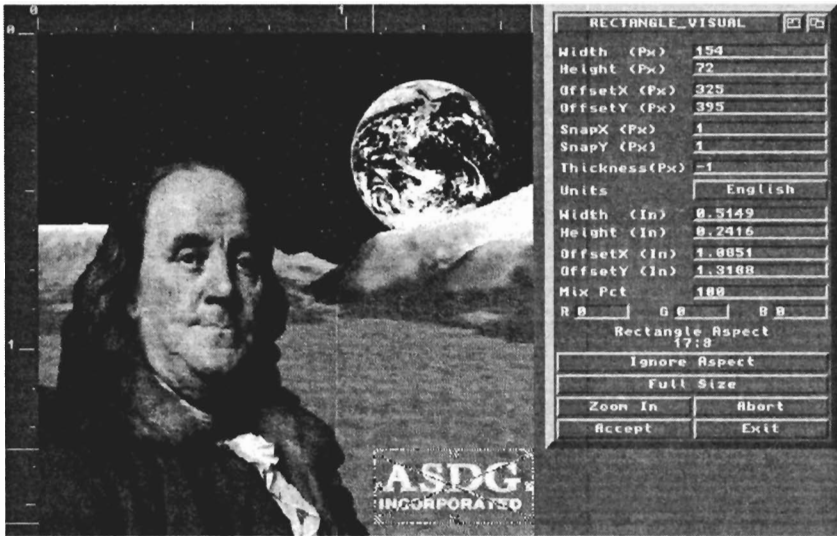


Figure 8.18: The Rectangle_Visual control screen and panel showing how a completely filled rectangle is represented.

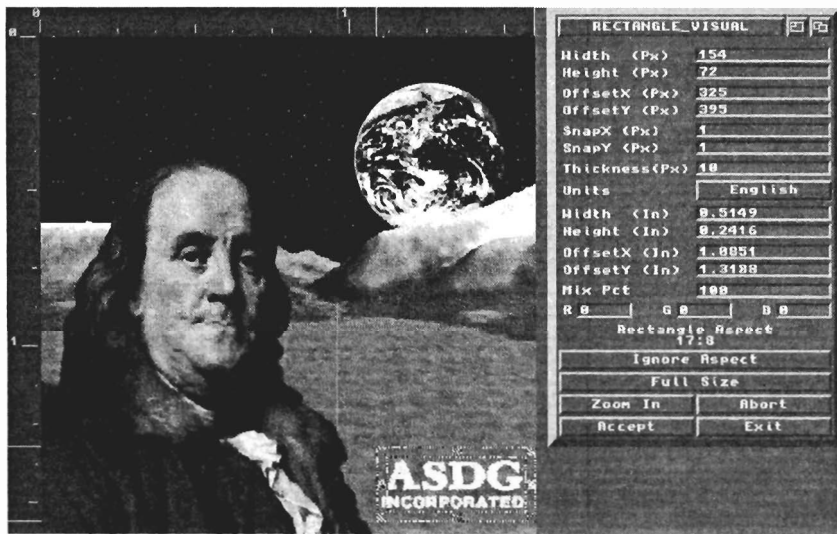


Figure 8.19: The Rectangle_Visual control screen and panel showing how a rectangle with a non-zero thickness (but not completely filled) is represented.

logo. `Rem_Isolated_Pxls` is less useful, though it can produce some artistic results, when used on color scans or other true color bitmaps.

In order to use `Rem_Isolated_Pxls`, you must have first rendered an image into 2, 4, 8, 16, 32, 64, 128, or 256 colors or grayscales. `Rem_Isolated_Pxls` does not operate on HAM images.

After rendering the image, selecting the `Rem_Isolated_Pxls` operator causes the operation to take place.

`Rem_Isolated_Pxls` operates directly on the rendered image data. Therefore, aborting a RIP in progress will result in a partially RIPped image.

8.33 `Rendered_To_Raw`

The `Rendered_To_Raw` operator converts the rendered data currently held within ADPro's memory back into the raw data it represents. In doing so, it replaces the raw data which was previously held in ADPro's memory. This operator makes it possible, for example, to save rendered data in file formats which only support raw data. Or, to send rendered data to devices whose savers only support raw data.

The operator can also be used for special effects. For example, if you wish to composite a true color object over a 16 color backdrop, you can render the 16 color backdrop, invoke this operator to convert the rendered 16 color data back into 24 bit-plane raw data, then composite in the true color foreground object.

NOTE Also, if you have the latest `dctv.library` (at least version 3.48) in your **LIBS:** directory, you can use this operator to decode DCTV images back to raw image data. Simply load in a DCTV encoded image using the IFF loader, then execute this operator. `Rendered_To_Raw` will detect that the currently loaded image is a DCTV image and will properly decode it back into raw data. The version of the `dctv.library` that allows for this conversion is included with this version of ADPro.

This operator can fail only if there is not enough memory available to hold the raw version of the rendered memory currently in ADPro's memory.

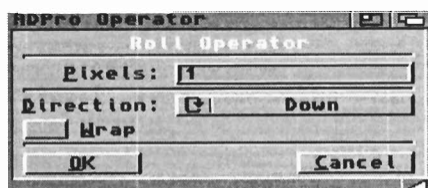


Figure 8.20: The Roll operator control panel.

8.34 Roll

The Roll operator (as shown in Figure 8.20) will allow you to create roll effects from within ADPro. This operator is useful in conjunction with the image composition facilities in ADPro to create animations where one picture shifts off screen while another image shifts on screen.

The **Wrap** checkbox allows you to specify whether or not the image will wrap as it rolls. If **Wrap** is checked, any portion of the image moved off the screen on one side, will appear in the vacated space on the other side. If it is not checked, the vacated space is filled with black.

The **Direction:** cycle gadget allows you to specify the direction in which the picture will be rolled.

The **Pixels:** input field tells the Roll operator by how many pixels you would like the image rolled. If you set this value to 100, the image will be rolled 100 pixels in the specified direction.

8.35 Rotate

With the Rotate operator, you can rotate circular areas of your image. The direction, center, amount, and quality of rotation, as well as the amount of blurring used at the edge of the circle, can be controlled.

Select the Rotate operator from the ADPro operator list. After selecting the operator, a preview screen should appear with the Rotate operator control panel on top of a grayscale representation of the currently loaded image.

The Rotate Circle visually describes the circular portion of your image that will be rotated. As you can see in Figure 8.22, the Rotate Circle is made up of a thin circle with a crosshair at its center, a smaller circle within this, a horizontal radius line in the right half of the circle, and a dashed line extended from the center of the circle.

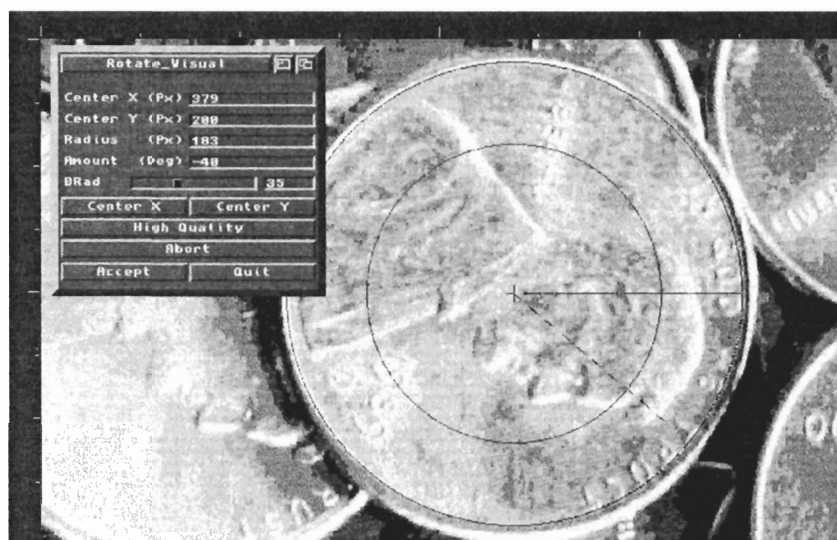


Figure 8.21: The Rotate operator control screen.

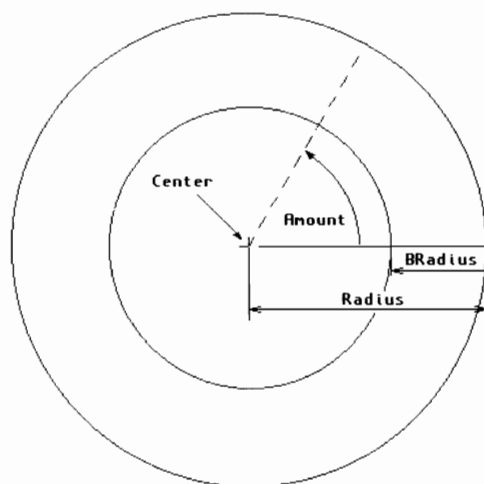


Figure 8.22: Components of the Rotate Circle.

Although values can be entered into the control panel's input fields, the more intuitive way of specifying the location of the rotate area on the image is to move the center crosshair on the Rotate Circle. Moving this crosshair moves the Circle as a whole around the image. To do this, place the mouse pointer above the crosshair, hold down the left mouse button, move the mouse so that the crosshair moves to the new location, and release the hold of the mouse button. Notice that when you move it around the image, the values in the **Center X (Px)** and **Center Y (Px)** input fields change as well. This gives you real-time feedback of your changes.

To center the Rotate Circle horizontally on your image, select the **Center X** button. To center it vertically, select the **Center Y** button.

Now that you've selected the center of your rotate area, you will want to specify the radius of this Circle. If you click the left mouse button while the pointer is anywhere but on the crosshair, the radius of the circle will snap to the new location. The actual radius (in pixels) is shown in the panel's **Radius (Px)** input field.

To define how much rotation will be done, you will have to enter a value in the **Amount (Deg)** input field. The current amount of rotation is shown on the Rotate Circle as a dashed line extending from the center of the Circle. An **Amount (Deg)** of 0 (no rotation) places the dashed line directly on top of the horizontal radius line. Positive **Amount (Deg)**s move the dashed line in a counter-clockwise direction. Negative **Amount (Deg)**s move it clockwise. Note that you cannot modify this dashed line's position with the mouse pointer.

The amount (percent) of blurring at the edge of the Rotate Circle can be controlled by the value specified in the **BRadius(0-99)** input field. This is represented as the innermost circle within the Rotate Circle. A value of **0** produces no blurring, whereas a value of **99** produces blurring across the entire radius of the Circle. The position of this inside circle, as well, cannot be modified with the mouse pointer.

The speed (and correspondingly, the quality) of the rotate operation can be controlled with the **Quality** button. Clicking on the button will toggle the type of quality between **Fast** (not the best quality, but faster to complete) and **High Quality** (high quality, but slower to complete).

To abort the display of the current image, select the **Abort** button. Only a portion of the entire image may be shown on screen, but you can still move the Rotate Circle almost anywhere on the image.

To accept the current Rotate Circle values and begin the operation, select the **Accept** button. The Rotate screen will disappear and a meter window will appear showing the progress of the operation. To abort the operation at any time, select the **Abort** button.

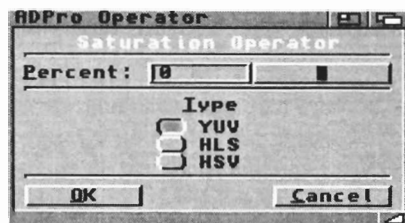


Figure 8.23: The Saturation operator control panel.

If you have finished modifying your image for this session and want to return to the ADPro window, select the **Quit** button.

8.36 Saturation

This operator (whose control panel is shown in Figure 8.23) allows you to adjust the amount of color saturation (using the **Percent**: slider and input field) in an image. If a negative value is given, this operator will dull the colors while a positive value will make the colors more intense. Specifying a value of zero will cause no change to the image.

Specifically, this operator will either move colors closer to or further away from gray depending upon the value you specify. The operator can base its computation on three different color models (available as **Type** radio buttons):

- YUV** In the YUV model, pixels will be moved closer to or further away from gray as defined by the Y or Luminance component of each pixel.
- HSV** In the HSV model, pixels will be moved closer to or further away from gray as defined by the V or Value component of each pixel.
- HLS** In the HLS model, pixels will be moved closer to or further away from gray as defined by the L or Lightness component of each pixel.

Note that since all computations are made within one of three color models, specifying the same adjustment will produce different results based upon which color model was set at the time of execution.

8.37 Scale

When 8 bit-plane gray or 24 bit-plane color data is present in memory, you may digitally reduce or enlarge the width or height (or both) of the image.

This is useful in a number of ways. For example:

- The individual pixels in an NTSC screen are not square. In fact, they are higher than they are wide in low resolution, non-interlaced displays. However, the pixels produced by many image sources such as scanners and 3D modelling programs are perfectly square. This mismatch in aspect ratio can result in NTSC images which appear vertically stretched. By reducing the height of an image to approximately 86 percent of its original size, you can produce images with an accurate aspect ratio for NTSC screens.¹
- Various artistic effects can be accomplished by reducing the width and height individually.
- You can create low resolution interlaced images easily by reducing the width alone by 50 percent and then enabling interlaced mode.
- You can create high resolution non-interlaced images (such as in the preparation of icons for the Workbench) by reducing the height alone by 50 percent.
- You can cause an image to fit into any desired standard screen size. You can preserve the image's aspect ratio by reducing both the width and height by the same amount. Or, you can reduce the width and height by different amounts.
- You can produce a special effect called "pixelization" or "posterization" by significantly reducing the image and then significantly enlarging it.

Selecting the Scale operator when raw data is available in memory will cause the scaling control panel (shown in Figure 8.24) to appear.

Using this control panel, you can specify a reduction or enlargement in three ways:

1. You can specify the number of pixels to be the resulting width and/or height using the supplied integer input fields.
2. You can specify the percent of reduction (or enlargement) of the width and/or height using the slider or input field.
3. You can select one of the percent preset buttons.

You can even select the **Lock Aspect** checkbox to force the pixel aspect to be the same. So, when you change one of the dimensions, the other dimension changes accordingly to retain the aspect. Note that the pixel aspect might not

¹Since the actual aspect ratio of a given screen will vary from monitor to monitor, the figure of reducing an image's height by 14 percent is *only approximate*. Also, PAL video is much closer to square than NTSC. Therefore, PAL screens may not require adjustment at all.

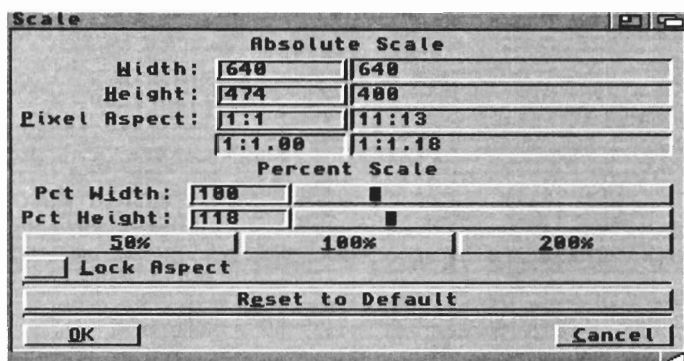


Figure 8.24: The Scale operator control panel.

stay locked to the exact value, especially when the image is being drastically resized. For the most part, however, it will try to choose a width and height that closely matches the aspect at the time the checkbox was checked.

To reset the values to what they were when the control panel first appeared, you can select the **Reset to Default** button. To ignore any scaling you might have specified, you can select the **Cancel** button.

Note that using the sliders, you can only specify reductions down to 25 percent of original size, or enlargements up to 400 percent of original size. However, by entering the scaling amounts manually, you can define reductions to less than 25 percent of original size, or enlargements to larger than 400 percent of original size.

To accept a set of reduction or enlargement values, simply select the **OK** button. *Unlike other operators, scaling operations are initiated immediately after acceptance.*

Quality digital scaling requires massive computation. The computation time of ADPro's scaling facility is proportional to the size of the resulting image. Also note that scaling permanently alters the raw data in memory. After an image has been scaled, you would have to reload from disk (if it wasn't saved to the TEMP buffer before scaling) in order to get back the unscaled data.

Pixel Aspect

The IFF format supports the storing of a pixel aspect ratio with the file. When ADPro loads an IFF-ILBM file containing this pixel aspect ratio specification, it presents this information in the Scale control panel as shown in Figure 8.24. The pixel aspect defaults to 1:1 (pronounced "1 to 1") when ADPro loads a

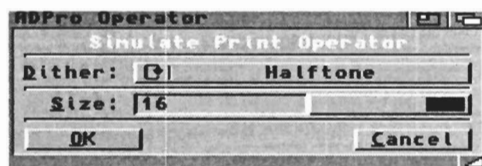


Figure 8.25: The Sim.Print operator control panel.

file which does not contain any pixel aspect information.

ADPro maintains the pixel aspect information as scaling operations take place. The aspect shown in the right column in Figure 8.24 represents the current pixel aspect. The aspect shown in the left column represents the pixel aspect which would result from the currently defined scaling operation if it were to be executed.

This information can be used to your advantage especially if you know that the device this picture is intended for display (or printing) has a specific pixel aspect. By adjusting the scaling controls so that the pixel aspect shown in the left column matches the intended pixel aspect of the device, the aspect of objects within the image will be preserved.

For more information about pixel aspect, please see Section 3.6.

8.38 Sim.Print

The Sim.Print (Simulate Print) operator allows you to simulate the printing process, using the current raw image in ADPro's memory as its printing "surface." Specifically, an image that would have normally been dithered (to produce higher color fidelity) for paper output would itself be modified with the dither. Unlike other operators, the processing will not affect the original raw image data—the rendered image will be the simulated print.

As shown in the control panel (Figure 8.25), you have control over the **Dither:** and **Size:**. The **Dither:** cycle gadget lets you select the dithering method to use on the image. You can select from one of **Halftone**, **Ordered**, **Vertical Line**, **Horizontal Line**, **Fwd Diagonal Line**, **Bwd Diagonal Line**, **Fwd Brick**, or **Bwd Brick**.

The **Size:** slider lets you select the size of the dither mask to use. You can choose from one of **2**, **4**, **8**, or **16**.

For a complete description of these dithering methods and mask sizes, see the PREFPRINTER saver section.

The **OK** button will perform the simulated print on your image. The **Cancel** button will let you exit this control panel without executing this operator.

NOTE Note that if your current display module is a 24-bit graphics board, the rendered image data will contain the “simulated print”. As such, selecting the **Execute** button will **not** display it. You will need to perform a `Rendered_To_Raw` operation to display it, as these display boards will show the raw image data if it exists.

8.39 Text_Visual

The `Text_Visual` operator allows you to merge text into the ADPro image buffer using a WYSIWYG user-interface. The text can be written using several special effects including: embossing, tinting, and anti-aliasing. And, if you are working under Workbench 2.0 or later, you can use scalable (CompuGraphic) fonts to create a font of any point size you desire.

When you execute the `Text_Visual` operator, a grayscale representation of the ADPro image buffer is displayed on your screen, along with the operator’s control panel. This panel can be used to enter your text string (or strings), choose a font type and size, control the placement of the text, and define which special effects will be used to draw the text.

The `Text_Visual` operator’s main control panel (shown in Figure 8.26) is laid out into two sections. The buttons and input fields on the left hand side of the panel are concerned with choosing and placing your text string on the screen, whereas the controls on the right hand side of the panel are used to determine how the text is drawn into ADPro’s image memory.

The buttons and input fields presented in the main control panel are described below.

The input field (labelled as **String:**) in the upper left hand corner of the control panel is where you enter the string you want to appear on the screen. For instance, let’s say you wanted to place the phrase, **Hi there!** on the screen. Click in the **String:** input field and enter the string **Hi there!**. After you press the **Return** key, the text you entered will be rendered internally within the `Text_Visual` operator. For large strings and especially with large fonts, this may take a few moments.

When the text has been assembled within the `Text_Visual` operator’s memory, it will be flashed on-screen. The on-screen representation of the text will have been scaled to properly reflect the relative size of the text compared to the size of the image you are working on.

Note that you may have left the previous text positioned under the space now

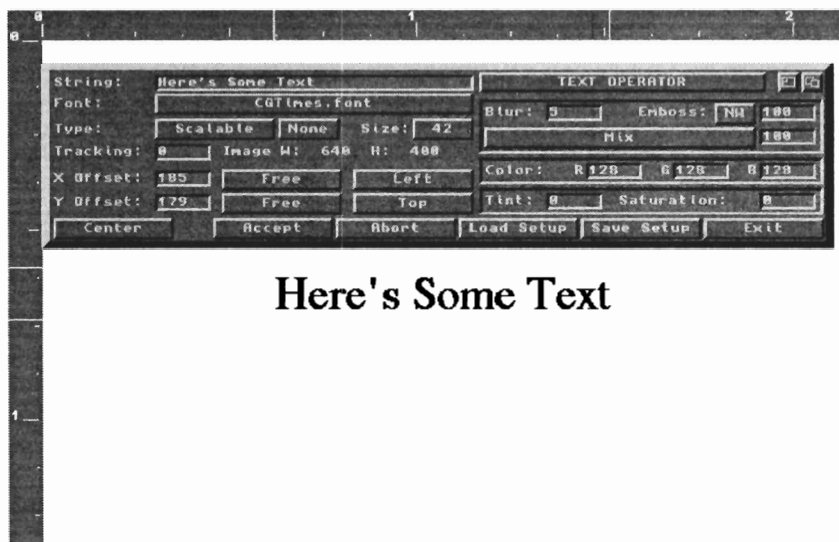


Figure 8.26: The Text_Visual operator control screen and main panel.

occupied by the control panel itself. You can tell when the text is finished being assembled for previewing when your mouse pointer assumes its default shape (after turning into the busy clock shape) again.

Directly below the **String:** input field are the closely related **Font**, **Type**, **Style**, and **Size** controls. Clicking on the **Font** button will bring up a list of all the available fonts on your system. If you have specifically requested to use scalable fonts, only the fonts whose name ends with **.otag** are allowed to be selected (since only these are outline fonts). If you have selected to use bit-mapped fonts, these will be presented along with those scalable fonts which have predefined bit-mapped sizes.

The **Type** rocker switch allows you to toggle between bit-mapped and (should you have them) scalable fonts. The scalable font option is available only if you are running under AmigaOS 2.04 or later.

When switching to scalable fonts or when changing the size of a scalable font, there is often a lengthy pause while the operating system performs the calculations it needs to convert a scalable outline into a bit-map ready for use. Consult the scalable font documentation from Commodore concerning ways you can decrease the length of this pause (by manipulating the font cache size, for example).

The **Style** button allows you to choose various attributes for the text you wish to use. These attributes include **Bold**, **Italic**, **Extended**, and **Underlined**

styles. Combinations of these styles are also permitted.

Note that font styles are handled by the operating system, not by ADPro. Some styles may not work properly with some fonts, especially with outline fonts. Also note that changing the style of a scalable font will often mean that the operating system must take some time to build the new type face from its scalable version.

The **Size** button allows you to choose the point size for the font you're currently using. If you are using a bit-mapped font, the size you pick must be one of the pre-existing point sizes. If you are using a scalable font, you can pick any size between 4 and 127 points. Note that if a pre-built bit-mapped version of the scalable font you're using doesn't exist, the operating system will take some amount of time to create it.

The **Tracking:** input field allows you to control inter-character spacing (the space between individual characters in your text string). Tracking can be used to improve readability or for artistic purposes.

Experiment with how this affects the look of your string by typing in various tracking values (try comparing zero tracking with a tracking of ten, for example).

Positive values expand the text string, negative values move the characters closer together.

Assuming you've entered a string (and pressed the **Return** key) and chosen a font, you can now move the rendered string around the screen by grabbing it with the mouse (while holding down the left mouse button).

This brings us to the **X Offset:** and **Y Offset:** input fields. These input fields were designed to aid you in placing your text string on-screen.

The **X Offset:** and **Y Offset:** input fields each have three boxes to their right. The first box to the right of each input field displays the coordinates of the text string on-screen. You can directly enter the screen coordinates in these input fields to position your text on-screen or, as mentioned before, you can use the mouse to position your text. If you use the mouse to position your text, the input fields immediately to the right of **X Offset:** and **Y Offset:** will be updated continually as long as you have the left mouse button depressed.

Let's skip to the third box to the right of the **Offset** input fields. These rocker switches determine the point on your string from which the screen coordinates are derived. Think of these as defining the "handle" you use to position the text on-screen. The **X Handle** rocker switch allows you to choose from the **Left**, **Center**, and **Right** of the string; the **Y Handle** rocker switch allows you to choose from the **Top**, **Baseline**, and **Bottom** of the string.

The ability to define where the "handles" are placed make it easy to create left

or right justified text, and horizontally aligned or vertically centered text.

Finally, the rocker switches immediately to the right of the **Offset** input fields restrict the movement of the text on-screen to help you line things up. These rocker switches have two states: **Free** and **Locked**.

Setting one of the switches to **Locked** freezes the axis so that no more movement along that axis is allowed. As an example, you might use the mouse to position your text to the right height. Then, flip the **Y Offset** rocker switch to its **Locked** state. Then, if you press the **Center** button, the text will be centered based only upon the X axis (width) and not along the Y axis (height).

The **Center** button (at the lower-left hand corner of the Text.Visual operator control panel) allows you to immediately center your text string on-screen. The **X Offset** and **Y Offset** rocker switches determine how the string is centered. If both **Offset** rocker switches are set to **Free**, then the string is centered in both X and Y (at the dead center of the screen) axes. If one of the axes is **Locked**, the string will be centered along the unlocked axis.

We now move on to the rocker switches and input fields on the right hand side of the Text.Visual operator control panel. These controls describe the appearance of the text, and how it will be drawn into the existing image in memory.

The effect of all of these controls are additive, so a extremely large number of special effects are possible. Experimentation is essential and rewarding.

The **Blur**: setting is used to control how much smoothing (or anti-aliasing) will be applied to the edges of each character in the text string. By changing the blur value, you can control how jagged or fuzzy the edges of each character will appear. Valid values are between -1 and 16.

A value of -1 disables blurring, so the characters will be drawn exactly as they are defined. As in the Blur operator (see Section 8.3), a value of 0 gives maximal blurring. A value of 16 gives minimal blurring.

The **Emboss**: rocker switch gives you the ability to give each character a three dimensional appearance. You can make your text appear as though it was punched into (setting **IN**) or is protruding from (setting **OUT**) its background.

Or, you can specify the **Emboss**: to be from one of the eight cardinal compass directions (**N**, **NE**, **E**, **SE**, **S**, **SW**, **W**, or **NW**).

The input field to the right of the **Emboss**: rocker switch holds a number which is used to control the "amount" of embossing that occurs. Valid values for this number are between 0-100. As this number grows larger, the text will appear more embossed, that is it will appear to stick out of (or into) the picture more.

Directly below the **Blur:** and **Emboss:** controls is the button which is used to control how your text is overlayed (or drawn) into the screen. Clicking on this button causes it to cycle through one of five possible values: **Mix**, **Lighten**, **Darken**, **Transparent**, and **Outline**.

Immediately to the right of this button is an input field which can hold a number between 0-100 (as with the **Emboss:** button). This field controls how intensely each of the above mentioned options will take effect.

- Selecting **Mix** causes your text (and the color you have chosen for it) to be mixed with the underlying ADPro bitmap. The value in the intensity input field controls how much color from the background shows through. A value of 0 causes the background to show through entirely; a value of 100 causes your text to completely overwrite the picture below it.
- Selecting **Lighten** causes ADPro to lighten the image beneath your text without changing that area's hue. Since **Lighten** takes its coloring information from the hues already found in the image, the values you specify in the **Color:** input fields are ignored.

The **Intensity** input field controls how much lightening occurs. A value of 0 would do nothing (no text would be seen), and 100 will cause the greatest effect.

- Conversely, **Darken** causes ADPro to darken the image beneath your text. As with **Lighten**, the values you specify in the **Color:** input fields are ignored since the colors used in the darkening operation are those already contained in the image.
- The **Transparent** choice turns off all direct mixing of the text. Other drawing, such as that caused by the embossing effects, will still be visible.
- The **Outline** option causes only the outline of the text to be drawn. Again, the **Intensity** value controls how much of the background beneath the text shows through. In order for **Outline** to work, blurring must be enabled.

The **Color:** control area contains three input fields. This is where you enter the desired color for your text string. Color is input in the 24 bit RGB space; valid values are between 0 and 255.

Directly below the **Color:** input fields lie the **Tint:** and **Saturation:** input fields. These two fields are used to give technical users more control over the appearance of their text. Unlike most operators in ADPro, which work in RGB (Red, Green, Blue) color space, these two input fields affect the color of your text in HSV (Hue, Saturation, Value) color space. While a detailed explanation of color theory is beyond the scope of this manual, here is a brief explanation of the function of these fields.

- The **Tint:** field is used to control how strongly the RGB color you specify in the control panel will tint the pixels beneath the text. At a

setting of 100, the pixel beneath the text will take on the RGB color you specify, completely. At a value of 0, the pixels beneath the text will be left unchanged. This function changes the hue of the pixels, but leaves alone their saturation and value.

- The **Saturation:** field works in the following way: The operator will compute the color saturation of the RGB color you specify in the control panel. At a setting of 100, the pixels beneath the text will assume completely the saturation of your RGB color. At a setting of 0, the pixels beneath the text will be unchanged. This function changes the saturation of pixels, but leaves alone their hue and value.

Once you have all the settings defined, you can press the **Accept** button to actually overlay your text with the ADPro bitmap. ADPro draws a box around each character on-screen as it is being processed. By watching the progress of this outline box as it travels across your text, you can follow the progress of the text operation.

Once it has finished, you can exit the Text_Visual operator (by clicking on the **Exit** button), or you can set up another line of text.

Clearly, there are a considerable number of possible effects and combinations of effects. For this reason, the **Load Setup** and **Save Setup** buttons allow to store favored settings on disk and recall them when desired.

8.39.1 Example Usage

Load an image of your choice into ADPro. Next, select the Text_Visual operator. After a slight delay, your image will be displayed in grayscale, along with the Text_Visual operator's control panel.

Click in the **String:** input field and type in a string of your choice (for example, **Hello World!**). Next, click on the **Type:** rocker switch so that it says **Bitmapped**, if it doesn't say that already.

Afterwards, click on the **Font:** button. This will bring up a list of the available fonts on your system. Most Amiga's should have the **emerald.font** font installed. Choose this font, with a point size of 20. (If you do not have this font, choose another font which you do have.)

After this, click in the **Tracking:** input field. Enter a value of 20. Notice how this expands the text after you press the **Return** key on your keyboard.

Now move the mouse up to the text string. Experiment dragging the text around the screen. If both the **X Offset:** and **Y Offset** range of motion rocker switches are set **Free**, then you should be able to move the text around anywhere within the borders of your image.

Set the text down at some particular height. Click on the **Y Offset:** range of motion rocker switch so that it says **Locked**. Now press the **Center** button. Notice how the text moves so that it is horizontally centered on-screen but that its height on-screen did not change. This is because the **Y Offset:** range of motion was “locked.”

Now go over to the **Color:** input fields. In this case, set the Red (**R**) and Green (**G**) values to 255, and the Blue (**B**) value to 0. This will cause the text to be rendered in yellow.

Set a value of 8 into the **Blur:** input field and click on the **Emboss:** rocker switch until it displays **NW** (Emboss the text from the NorthWest). In the input field immediately to the right of the box displaying **NW** enter a value of 80.

Click on the long button immediately below the **Blur:** and **Emboss:** controls. Repeatedly click on it to cycle through the various options until the **Mix** setting is displayed. Now enter a value of 70 for the mix level (the input field just to the right of the **Mix** rocker switch).

Select the **Accept** button. ADPro will draw a box around each character in the string as it is drawn into the picture. When the operator is finished, quit it by selecting the **Exit** button.

From the main ADPro control window, select the **Execute** button to see your changes in full color.

8.40 Tile

The Tile operator allows you to specify a rectangular area which will be repeated throughout the currently defined bit-map to create a “wall paper” or tiled effect. You can specify a vertical or horizontal offset which will skew the tiling either vertically or horizontally.

NOTE Use of this operator from the user interface is no longer suggested. Instead, use the Tile.Visual operator which provides you with an on-screen representation of the operation, rather than the numeric-only user interface which this operator presents. This operator is retained for use through ARexx control where the visual capabilities of Tile.Visual will not be used. See Section 8.41 for more information about the Tile.Visual operator.

The Tile operator’s control panel is shown in Figure 8.27. Specify the left edge (pixel column number) of the rectangle you wish to tile in the input field marked **X Offset:**. Similarly, specify the top edge (pixel row number) of the

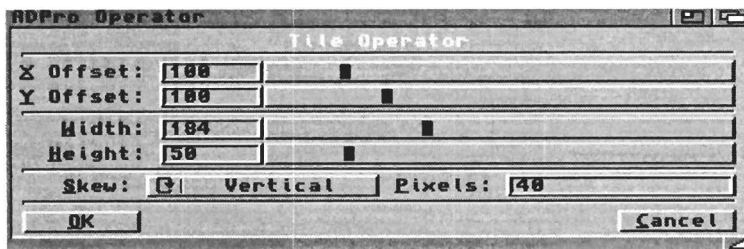


Figure 8.27: The Tile operator control panel.

rectangle you wish to tile in the field marked **Y Offset:**. Also specify the width and height of the rectangular area (in pixels) in the fields marked **Width:** and **Height:**, respectively.

The **Skew:** cycle gadget can be set to either **Vertical** or **Horizontal**. The amount of skew is specified in the **Pixels:** integer input field.

Selecting the **OK** button causes the tiling operation to actually take place. Selecting **Cancel** exits the Tile operator without actually performing a tiling operation.

Figure 8.28 shows a 320 by 200 bit-map which has been tiled with a 32 by 20 pixel rectangle (shown in gray) with no skew. A no-skew tiling is characterized by the rigid grid pattern as Figure 8.28 shows. You can specify a no-skew tiling by setting the skew amount to 0 (the skew type may be set to either vertical or horizontal).

Figure 8.29 shows a 320 by 200 bit-map which has been horizontally skewed using the same 32 by 20 pixel rectangle (shown in gray). A horizontal skew is one in which the tiled rectangles all line up horizontally.

Finally, Figure 8.30 shows the same 320 by 200 bit-map which has been vertically tiled by the same 32 by 20 pixel rectangle (shown in gray). A vertical skew is one in which the tiled rectangles all line up vertically.

Notice that the source of the tiling operation (shown in gray in Figures 8.28, 8.29, and 8.30) remains the same. This shows that you can repetitively execute the Tile operator over the same rectangle, varying the skew amount and type, as long as you keep the same width, height, left edge and top edge.

Also notice that the tiled area did not fit evenly into the total bit-map. That is, fractional portions of the tiled area can be found along the edges of the bit-map. This will necessarily happen when the width or height of the tiled area does not evenly divide into the width or height of the entire bit-map. You can also force this to happen to offsetting the placement of the tiled area from the top left corner of the bit-map.

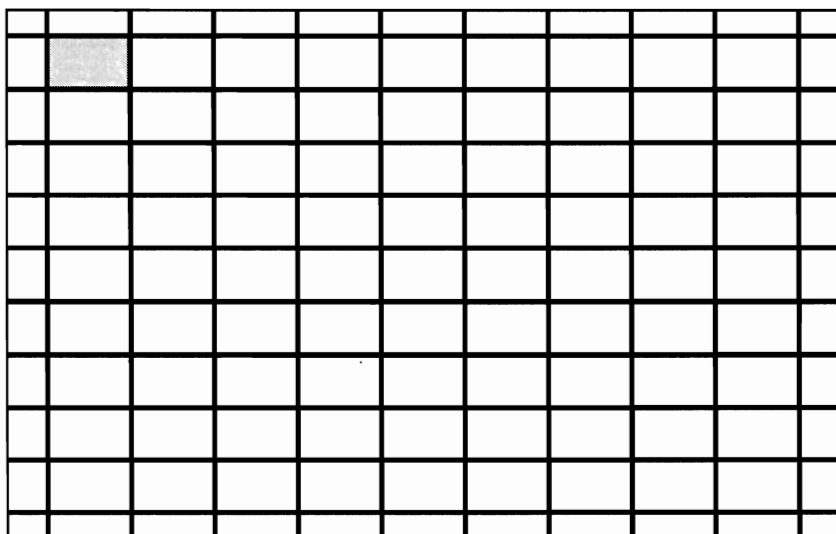


Figure 8.28: A tiling showing no skew.

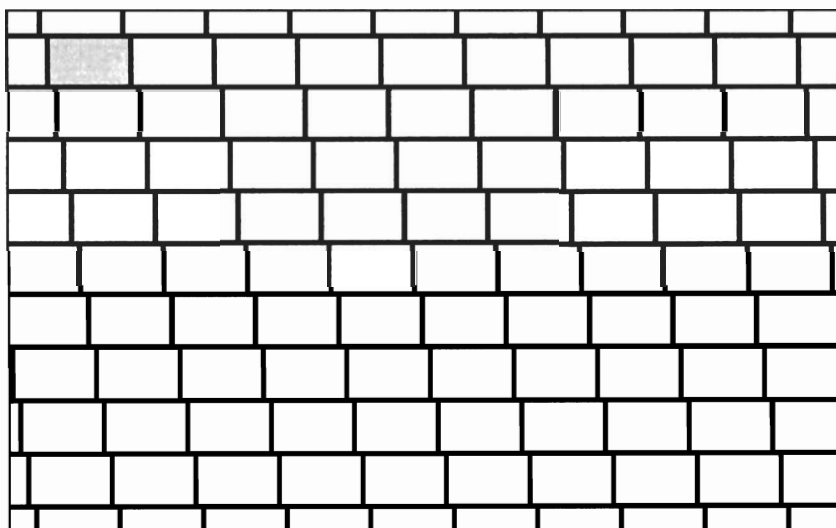


Figure 8.29: A tiling showing horizontal skew.

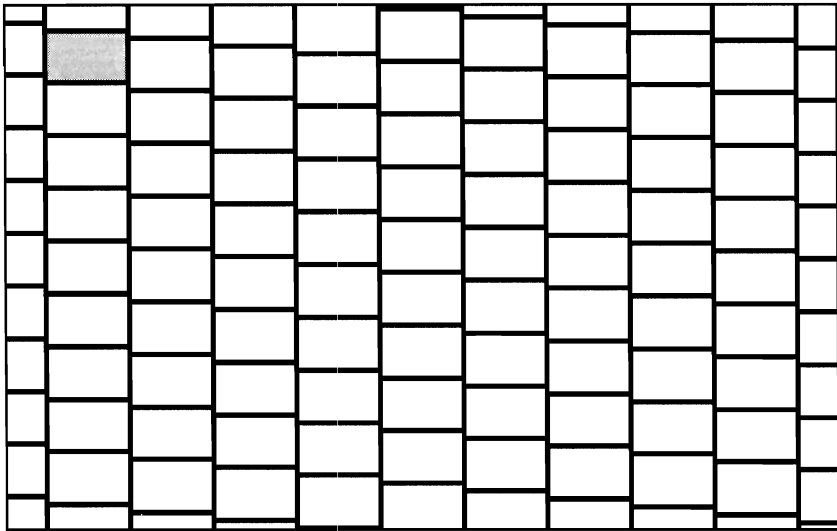


Figure 8.30: A tiling showing vertical skew.

In the examples shown in the preceeding figures, the tiled area was 32 by 20 pixels in size. The entire bit-map was 320 by 200 pixels in size. Clearly, by specifying a left edge and top edge of 0, the tiled area would evenly fit in the larger bit-map. This is shown in Figure 8.31. However, if you wish to force the tiled area to not fit evenly in the larger bit-map, you can reposition the source rectangle as needed.

For example, if you wished to tile a 320 by 200 pixel bit-map with the 32 by 20 pixel rectangle starting at a left and top edge of 0, but wanted to force a fractional part along the borders of the bit-map, you might perform the following steps:

1. Using either the `Crop_Image` or `Crop_Visual` operator, isolate just the area you wish to tile from. Save this area to a temporary file.
2. Using the `BACKDROP` loader, create a bit-map of the desired size.
3. Using image compositing, load the area to be tiled from the temporary file at an offset which will yield the tiling you wish.
4. Tile the image, specifying a width and height of the area you are tiling from and left edge and top edge from where you loaded the area in the previous step.

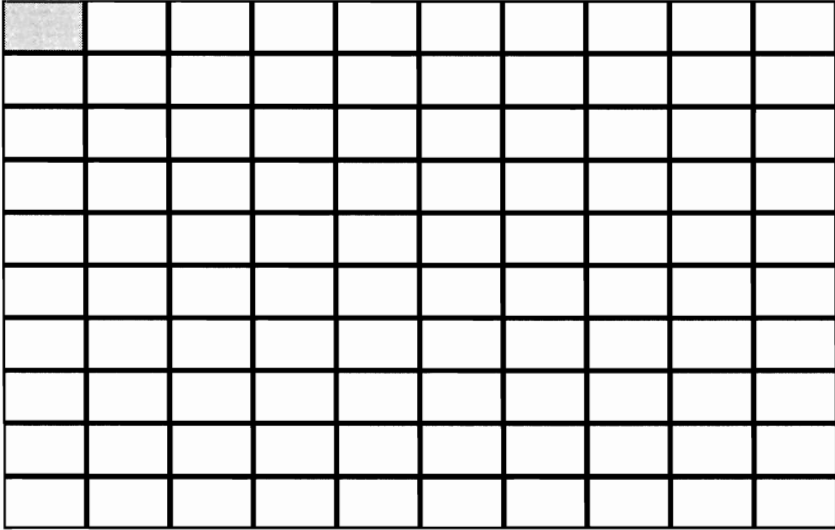


Figure 8.31: A tiling showing no skew in which the tiled area evenly fits in the larger bit-map.

8.41 Tile_Visual

This operator provides an on-screen WYSIWYG user interface (shown in Figure 8.32) to the tiling operation. We suggest that you use this operator rather than the Tile operator when working with ADPro from its user interface. However, when working through ARexx control, we suggest that you continue to use the Tile operator since its small size will allow it to load from disk more quickly than Tile_Visual. Also, from ARexx, the visual benefits of Tile_Visual are negated.

The box representing the area-of-interest indicates where the bounds of the area which will be tiled. You can use the buttons marked **Full Size** and **Zoom In** to see finer or grosser detail. The **Accept** button will cause the tiling operation to actually take place. The **Exit** button will leave the Tile_Visual operator without performing a tiling operation.

Pressing **Shift + Return** on your keyboard is the same as selecting the **Accept** button. When the control panel is zipped (it is shown “iconified”), pressing the **z** key on your keyboard is the same as hitting the **Zoom In** button. Also, when the panel is zipped, pressing the **u** key will unzip the panel (make it full size again).

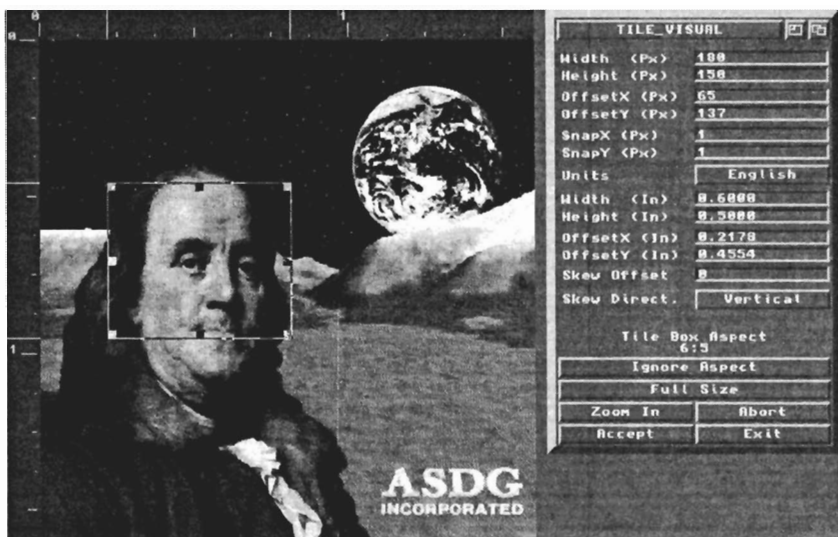


Figure 8.32: The Tile_Visual operator control screen and panel.

8.42 TPort_Controller

Selecting the Transport_Controller operator causes ADPro to look for a running copy of MicroIllusions' Transport Controller. If found, ADPro will communicate with the Transport Controller to record the Amiga displayable image currently in memory.

This operator requires the presence of rendered image data in Amiga displayable format. It will also tell you if it cannot locate a currently running Transport Controller.

Control will return to ADPro when the Transport Controller is through recording the image. The number of frames to be recorded is determined by the default value maintained by the Transport Controller. Direct control (from within ADPro) over the number of frames to be recorded is provided only via ARexx since the Transport Controller provides a user interface of its own.

8.43 Twirl

With the Twirl operator, you can twirl circular areas of your image. The direction, center, amount, and quality of twirl, as well as the amount of blurring used at the edge of the circle, can be controlled.

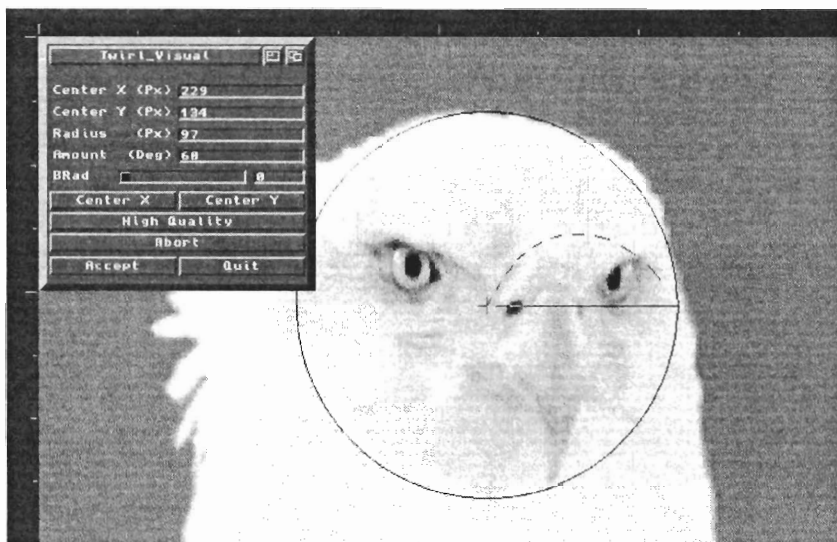


Figure 8.33: The Twirl operator control screen.

Select the Twirl operator from the ADPro operator list. After selecting the operator, a preview screen should appear with the Twirl operator control panel on top of a grayscale representation of the currently loaded image.

The Twirl Circle visually describes the circular portion of your image that will be “twirled”. As you can see in Figure 8.34, the Twirl Circle is comprised of a thin circle with a crosshair at its center, a smaller circle within it, a horizontal radius line in the right half of the circle, and an dashed arc extending from the center of the circle out to the edge.

Although you can enter values into the control panel’s input fields, the more intuitive way of specifying the location of this twirl area on the image is to move the center crosshair on the Twirl Circle. Moving this crosshair moves the Circle as a whole around the image. To do this, place the mouse pointer above the crosshair, hold down the left mouse button, move the mouse so that the crosshair moves to the new location, and release the hold of the mouse button. Notice that when you move it around the image, the values in the **Center X (Px)** and **Center Y (Px)** input fields change as well. This gives you real-time feedback of your changes.

To center the horizontal position of the Twirl Circle, select the **Center X** button. To center the vertical position, select the **Center Y** button.

Now that you’ve selected the center of your twirl area, you will want to specify the radius of this Circle. If you click the left mouse button while the pointer

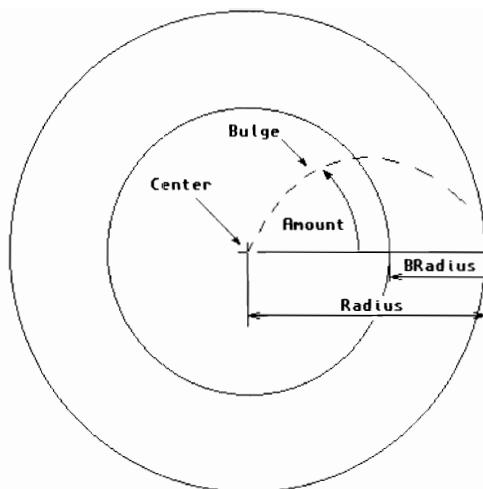


Figure 8.34: Components of the Twirl Circle.

is anywhere but on the crosshair, the radius of the circle will snap to the new location. The actual radius (in pixels) is shown in the panel's **Radius (Px)** input field.

To define how much twirling will be done, you will have to enter a value in the **Amount (Deg)** input field. The current amount of twirl is shown on the Twirl Circle as a “bulge” or arc from the radius line in the direction the twirl will take. This bulge also shows in what direction (clockwise or counter-clockwise) the area defined by the Twirl Circle will be twirled and by how much (in degrees). If a positive **Amount (Deg)** is specified, the arc will bulge upward and the twirl will be in a counter-clockwise direction. If a negative **Amount (Deg)** is specified, the arc will bulge downward, producing a twirl in the clockwise direction. Note that you cannot modify the bulge with the mouse pointer.

The amount (percent) of blurring at the edge of the Twirl Circle can be controlled by the value specified in the **BRadius(0-99)** input field. This is represented as the innermost circle within the Twirl Circle. A value of **0** produces no blurring, whereas a value of **99** produces blurring across the entire radius of the Circle. The position of this inside circle, as well, cannot be modified with the mouse pointer.

The speed (and correspondingly, the quality) of the twirl operation can be controlled with the **Quality** button. Clicking on the button will toggle the type of quality between **Fast** (not the best quality, but faster to complete) and **High Quality** (high quality, but slower to complete).

To abort the display of the current image, select the **Abort** button. Only a portion of the entire image may be shown on screen, but you can still move the Twirl Circle almost anywhere on the image.

To accept the current Twirl Circle values and begin the operation, select the **Accept** button. The Twirl screen will disappear and a progress meter window will appear showing the completeness of the operation. To abort the operation at any time, select the **Abort** button.

If you have finished modifying your image for this session and want to return to the ADPro window, select the **Quit** button.

8.44 Vertical_Flip

Many applications require the production of mirror images of the original data. Specifically, this is required by screen and heat-transfer printers. Selecting the Vertical_Flip operator will vertically flip the raw image data currently in memory.

Note that performing a horizontal flip followed by a vertical flip is the same as rotating the image 180 degrees. This can be used in conjunction with the Orientation controls (**No Composition**, which will do a Portrait load, and **Landscape**) to rotate images through 0, 90, 180, and 270 degrees. Note, as well, that you can also do these same rotations with the Rotate operator.

8.45 _ColorCharcoal

This pseudo operator applies a colored charcoal effect on your image. This effect might not work as well on some images as it does on others.

8.46 _DoubleSize

This pseudo operator doubles the width and height of your image. It is, in effect, the opposite of the Halve operator.

8.47 _Emboss

This pseudo operator will create an embossed version of your image. You can specify whether to emboss inward or outward.

8.48 `_Fresco`

This pseudo operator will create an fresco version of your image.

8.49 `_Highlight`

This pseudo operator will take a rectangular area and dim the image outside of this area, thereby “highlighting” the rectangular area.

8.50 `_Mirror`

This pseudo operator will allow you to mirror either the top, bottom, left, or right halves of your image.

8.51 `_OilPaint`

This pseudo operator will make your image looks as if it were done as an oil painting.

8.52 `_RemapNRrender`

This pseudo operator will render your image using an externally-saved palette file.

8.53 `_RotateImage`

This pseudo operator will rotate your image in 90, 180, and 270 degree increments.

8.54 `_ScaleToPixelAspect`

This pseudo operator will scale your image to a specific pixel aspect. This is very useful when transferring images between platforms and you want to guarantee that it will look similar on a display with a different pixel aspect.

8.55 `_Solarize`

This pseudo operator will do a simple solarization of your image.

Chapter 9

Displays

This chapter provides descriptions of the use and operation of all display modules accessible from ADPro. ARexx control of these display modules is described in Chapter 12.

9.1 Amiga

You can render to any native Amiga screen using the Amiga display module.

This display option will appear in the Set Render Screen requester as entries in the following form:

monitor_type:mode_description

A typical example is: “NTSC:High Res”.

If you do not see any entries of this form, then you probably do not have any monitor icons installed in the **Monitors** drawer. Use the following as a guide:

- Under Workbench 2.04, move the appropriate monitor icons from the **MonitorStore** drawer to the **Monitors** drawer of the system partition.
- Under Workbench 2.1 and later, move the appropriate monitor icons from the **SYS:Storage/Monitors** drawer to the **SYS:Devs/Monitors** drawer.

To use newly installed monitor types, either reboot the machine or double-click on each installed icon. The next time you display the Set Render Screen

requester, the screen modes defined by these monitor icons will appear.

If you see the text “No name for mode”, then the image just loaded did not contain a screen mode identifier or that identifier does not have a name. Simply select a named mode from the Set Render Screen requester.

9.1.1 Render Controls

If the **Width:** and **Height:** you specify is larger than the render screen's default dimensions, the screen that will open will be auto-scrollable—by moving the mouse pointer to the edges of the visible area, the rendered image will automatically scroll to show those parts of the image.

NOTE If you click the left mouse button on the render screen, you will return to the ADPro window. If you then click on the front/back gadget of the screen on which ADPro's window is located, you will be returned to the render screen. To cycle through the other open screens, click the left mouse button while the mouse pointer is in the upper-right corner of the render screen.

Note that if the render screen is larger than the visible area (i.e., when auto-scroll is active), you will have to scroll to the right-most part of the *screen* and not just the visible area to flip screens.

The keyboard sequence of **Left-Amiga + M** will always cycle screens.

9.1.2 Display Restrictions

To display your 8 or 24-bit image data in standard Amiga modes, you must create rendered image data in an Amiga displayable screen format. Due to the standard Amiga's graphics display limitation, you cannot view these 8 or 24-bit images directly on your Amiga monitor. For most current Amigas, the most colors you can display is 4,096 (HAM mode). Amigas installed with the Advanced Graphics Architecture (AGA) chip set will be able to display more colors. Consult Table 9.1 for a complete listing of available colors in various screen modes.

To create rendered image data, press the **Execute** button. The raw image data will be analyzed and the image will start to render (display), assuming you have selected your screen controls properly.

Setting	Colors	Bitplanes	Low Res	High Res
2	2	1	Y	Y
4	4	2	Y	Y
8	8	3	Y	Y
16	16	4	Y	Y
32	32	5	Y	A
64	64	6	A	A
128	128	7	A	A
256	256	8	A	A
EHB	64	6	Y	A
HAM	4096	6	Y	A
HAM8	262144	8	A	A

Table 9.1: Color modes and other information. The mixtures of color and display modes marked “A” are possible but require non-standard hardware or an AGA-equipped machine. ADPro allows these modes to be computed but will display them only if the appropriate enhanced hardware is available.

9.2 EGS

You can render to any display mode supported by any EGS-compliant graphics board (currently, the Ingenieurburo Helfrich’s Piccolo and RainbowIII, GVP’s EGS-28/24 Spectrum, and DKB’s Talon) using the EGS display module. The EGS Device Independent Graphics System is copyright by VIONA-Development.

This display option will appear in the Set Render Screen requester as entries in one of the following forms:

```
LEGS:mode screen_dimensions
LEGS:mode screen_dimensionsxdepth
Window on:EGS-DEFAULT Screen
```

The screen mode prefix (“LEGS:” in the above cases) may be different, depending upon which EGS board and how many boards you have installed.

The first form allows you to render color-mapped (2 through 256 color) images to an EGS screen. This form will actually appear in the list even if the EGS display module isn’t installed, as these modes would appear in the Amiga Display Database.

NOTE If the **Available Modes Only** option is not selected, Amiga-only

modes such as EHB, HAM, and HAM-8 will be available, but not applicable. If you intend to always use these EGS display modes, **Available Modes Only** should probably be selected.

The second form is similar to the first form, except that high color (15 and 16 bit) and true color (24 bit) data will be displayed.

The third form allows you to display the image in a scrollable window on the default EGS screen. If the screen isn't currently open, it will be opened.

NOTE This third form may not appear in the same part of the list as the Window display module modes if you have other graphics board display modules.

9.2.1 Render Controls

You can scroll and toggle the display as with the other display options. There are no EGS-specific display controls.

If you are displaying to a window on the default EGS screen, click on the left mouse button and drag the mouse to scroll the image; you can also use the cursor keys to scroll. Note that once the mouse pointer moves outside of the window, scrolling will stop. Click on the window's close gadget or press the **Esc** key to close the window and return to the ADPro controls. If there are no other windows open on the default screen, the screen will also close. Otherwise, it will stay open.

9.3 FC24

You can render to any display mode supported by Impulse's FireCracker24 using the FC24 display module.

This display option will appear in the Set Render Screen requester in the following form:

FC24:Board *#board_number* - *screen_dimensions*

NOTE The FC24 display modes can only display the current raw image data. To display the rendered image data, first switch to an Amiga display mode and render the image in 2 through 256 colors. Then, perform a **Rendered_To_Raw** operation. As this will overwrite your original raw data, you might want to invoke the TEMP saver before doing so.

9.3.1 Render Controls

When rendering an image, a small control window will appear on the same screen as ADPro. This is required so that you can use the cursor keys to scroll the image, if the image is larger than the visible screen area. This window must be active for the image to be scrolled.

Although the **Spacebar** will toggle between this small window and the ADPro window, the only ways to close this small window is to either click on its close gadget or press the **Esc** key.

9.4 OpalVision

You can render to any display mode supported by Opal Technology's OpalVision graphics board using the OpalVision display module.

This display option will appear in the Set Render Screen requester in one of the following forms:

```
OpalNTSC Palette Map:  mode_description
OpalNTSC True Color:  mode_description
OpalPAL Palette Map:  mode_description
OpalPAL True Color:   mode_description
```

9.4.1 Render Controls

When an image is displayed and the image is larger than the visible area, you can use the cursor keys to scroll around the image. These are the standard scrolling controls, as available with the other display modes.

9.5 Picasso

You can render to any display mode supported by Village Tronic's Picasso II graphics board (limited by your specific board configuration) using the Picasso display module.

This display option will appear in the Set Render Screen requester as entries in one of the following forms:

```
PICASSO: screen_dimensions
TRUECOLOR: screen_dimensions xsdepth
```

The first form limits you to color mapped (i.e., 2 through 256 color) images. Use the **Colors:** slider to select the number of colors.

NOTE If the **Available Modes Only** option is not selected, Amiga-only modes such as EHB, HAM, and HAM-8 will be available, but not applicable. If you intend to always use these Picasso display modes, **Available Modes Only** should probably be selected.

The second form gives you the 15, 16, and 24 bit modes of the Picasso.

9.5.1 Render Controls

You can scroll and toggle the display as with the other display options. There are no Picasso-specific display controls.

9.6 Retina

You can render to any display mode supported by MacroSystem's Retina graphics board (limited by your specific board configuration) using the Retina display module.

This display option will appear in the Set Render Screen requester as entries in the following form:

number_of_bits Bit screen_dimensions horizontal_freq x vertical_freq

9.6.1 Render Controls

When rendering an image, a small control window will appear on the same screen as ADPro. This is required so that you can use the cursor keys to scroll the image, if the image is larger than the visible screen area. This window must be active for the image to be scrolled.

Although the **Spacebar** will toggle between this small window and the ADPro window, the only ways to close this small window is to either click on its close gadget or press the **Esc** key.

9.7 Window

You can render to a window on any public screen using the Window display module.

This display option will appear in the Set Render Screen requester as entries in the following form:

Window on:*public_screen_name:public_screen_dimensions*

An example of an entry is: “Window on:Fred:724x478”

9.7.1 Render Controls

By default, the **Width:** and **Height:** values will be the same as the *public_screen_dimensions*, although you can enter a different value. ADPro will try to open a window as large as is specified or as large as the image itself (if it is smaller than the specified size).

While it is rendering in the window you can resize the window but you cannot scroll the image. Once rendering has finished (or after you interrupt it) you can use the scroll bars and arrows in the window’s border to pan around the image.

While rendering, you can stop it in two ways:

- If the render window is active, clicking on the window’s close gadget or pressing the **Esc** key will stop the rendering. Clicking or pressing it again will close the render window.
- If the meter window is active (i.e., render window is not active), selecting the **Stop** button will stop the rendering.

ADPro will remember the last size of the render window and open up to that size the next time you render, unless you have changed the size in the Set Render Screen requester. Also note that if you change the depth (number of colors) of the public screen, ADPro’s render mode definition will not get updated as well, so you should reselect the render mode from the Set Render Screen requester.

NOTE If you are rendering in Custom Palette mode (i.e., the **Custom Palette** checkbox in the Palette control panel is checked), only the **Colors (Used):** and **Offset Color Zero:** values will be used—you will still be locked to the number and selection of colors in the public screen.

It is a good idea when working with a custom palette to avoid displaying in a window. This will avoid the problem of seeing an image which does not properly represent your custom palette settings.

Chapter 10

FRED

FRED is a visually oriented list manager. The lists it manages, called sequences, are comprised of image names and other information. Individual images (or frames) in a sequence are represented on-screen with accurately rendered icons.

FRED is an independently executable program which can be run in conjunction with ADPro.

FRED can instruct ADPro to process each frame in a sequence, providing an easy method of batch processing. FRED also provides an extensible ability to call special purpose drivers which can cause ADPro to generate animation effects automatically. You can even preview animations within FRED, before (for example) committing them to a single frame recorder.

FRED is a powerful tool for the 24 bit single animator. It also benefits those who would like to batch process unrelated pictures but feel they are not up to the task of writing the appropriate ARexx program.

NOTE ADPro does not need to be running at the time FRED is launched—only when processing of images needs to be performed.

10.1 Overview

This section gives a quick overview of starting, displaying information about, and exiting FRED.

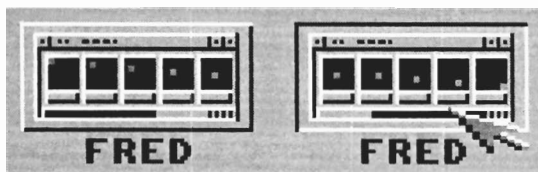
10.1.1 Starting the Program

You can start ADPro from either the Workbench or CLI/Shell.

Two sets of menus are accessible from within FRED. When the backdrop screen is active, a menu set including **Project**, **Settings**, **Tools**, and **AnimOps** items can be accessed. When a sequence window is active, a menu set including **Project**, **Edit**, **Animation**, and **Scripts** items can be accessed.

In both sets, the **Project** menu is the same.

Workbench Startup



For Workbench users, open the directory where the FRED program is located. Locate the left icon as shown above. It should have the name **FRED** below it. Double-click on this icon to start the program.

See Section 13.12 for more information on customizing FRED with tool type options.

Shell Startup

For Shell users, please start the ADPro program (if it isn't already running) before trying to start FRED. Note that you should **Run** the ADPro program if you want to start FRED from the same Shell.

Before starting the program, make sure you have previously executed the command:

Assign ADP_FRED: *location_of_FRED*

This is required for FRED to locate its support files. A good place to put this command is in your **S:user-startup** file, if it hasn't already been placed there by the installation utility.

From a Shell, type the following command (from the directory where FRED is located):

FRED

If FRED can initialize itself, then you will see its control screen. If no screen is displayed, you might have run out of available memory for the program to use. Note that FRED opens a 16 color high resolution interlace screen, which does require more memory to display than the ADPro screen (an 8 color low resolution non-interlace screen). If you are running on an AGA-equipped machine, this screen will open in 256 color mode.

If you are on an AGA machine but want to run FRED in 16 color mode to speed up the screen updating, use the **SHALLOW** command line switch. See Section 13.12 for more information on starting FRED with command line options.

10.1.2 Displaying Information About FRED

Selecting the **About...** menu item (under the **Project** menu) will display a requester with the following information:

- the revision of this copy of FRED
- text asking for your cooperating in keeping FRED professionally supported

10.1.3 Exiting the Program

To exit FRED, select the **Quit FRED** menu item under the **Project** menu.

10.2 Sequences, Cells, and Frames

Sequences, as mentioned earlier, are lists of images that will be processed by FRED. These sequences can be assembled, loaded, saved, processed, and animated.

10.2.1 Creating Sequences

Creating a new sequence is as easy as selecting the **New...** menu item (from the **Project** menu). An empty window, called the sequence window, will appear. Its title bar displays the name of the sequence (initially defined as

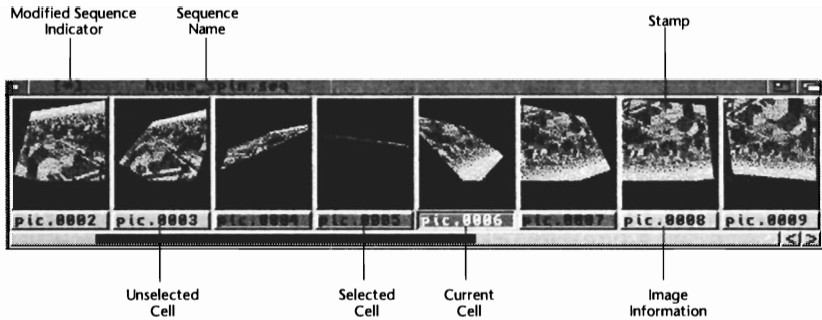


Figure 10.1: Components of a FRED sequence window.

`Untitled $x$ .seq`, where x is a number from 0 to 999), as well as if the sequence was modified at all ([*] for a modified sequence, [] if not modified).

These sequences are made up of cells (single images). These images can be similar to one another (such as cells of an animation) or a collection of varied images. What they share in common is the need to be processed. A sample sequence window containing cells (unselected, selected, and current) is shown in Figure 10.1.

10.2.2 Adding Images to a Sequence

Images can be inserted into a sequence as follows. Click on the sequence window to add to; the window should now be active. You then have the following options (as menu items under the **Insert** submenu of the **Edit** menu) for inserting images:

Image(s)... This option lets you choose one or more images (from the same directory) to insert into the currently active sequence. From the file requester that appears, you can multiply-select files by clicking on the first file, holding down the **Shift** key, then repetitively clicking on other files in that same directory. Once you are done, release the **Shift** key and click on the **OK** button.

Range... This option lets you insert a group of images that have numeric file extensions. For example, you can select to load `mypic.0004` (the first image in the range) through

`mypic.0020` (the last image) and FRED will automatically search for files with a base name of `mypic` and extensions of `.0004`, `.0005`, all the way up to `.0020`.

Note that the selected images will be inserted *after* the currently selected cell, which has its cell information box depressed and its text displayed in white. Also note that the numeric extension for the first image in the range does not have to be less than the last image's extension—it can be greater, if you want to insert images in reverse numeric order.

Directory...

This option lets you specify a directory from which to add all its files to the sequence.

Note that subdirectories are not examined for files nor are files necessarily inserted in increasing alphanumeric order—they are inserted in the order which they are found when reading the directory. To rearrange them, use the program's editing functions.

NOTE

Filenames that have embedded spaces within their name or path may cause problems when processing ARExx scripts, so please name your image files without any spaces and place them in directories which also don't have any spaces in their names.

10.2.3 Storing and Retrieving Sequences

If you want to save the defined cells of the sequence, select either the **Save** or **Save As...** menu item. **Save** will store the cells under its current sequence name (as shown in the window's title bar), if it has one. **Save As...** will let you save the sequence under a new name.

If no window is currently active (i.e., the backdrop screen is active), a list of currently open sequences will appear. If you select a sequence name and then press the **OK** button, that sequence will be saved.

To open a previously saved sequence, select the **Open...** menu item and select the sequence to load from the file requester that will appear.

You can have many sequence windows open on the FRED screen as graphics memory will allow. These windows can be hidden from view (also termed "iconified") by clicking on a window's zoom gadget.

To close a sequence window, select the window and choose the **Close...** menu item.

10.2.4 Switching Between Sequences

To make a window reappear or to activate another open window, select the **Reveal...** menu item. A list of currently open sequences will be displayed. If an 'I' appears before a sequence's name, the window corresponding to that sequence is iconified. Select the sequence to switch to and press the **OK** button. You can also double-click on the sequence name to bypass the **OK** button altogether.

10.2.5 Differentiating Cells and Frames

An important distinction in terminology must be made at this point before you continue reading the rest of the FRED documentation.

So far, we have termed the individual images of a sequence as "cells". What hasn't been discussed (but what will be later) is that each cell does not have to be a single instance of an image—it can represent the same image as if it had been entered as successive cells, *without having to insert them*. This is accomplished through the use of "frames."

For example, say that you want a particular cell to represent five instances of an image. Instead of adding five cells to the sequence, you can add just one and specify a length (in the **Length:** input field of the Image Information control panel) of 5. This ability comes in handy when animating the selected cells and batch processing the images.

10.3 Cell Selection and Editing

The cells in a sequence can be selected in many ways. The order of cells can also be modified. This section describes both operations.

10.3.1 Selecting and Unselecting Cells

Any cell in a sequence can either be selected or unselected. Any number of cells can be in the selected range. The currently selected (active) cell has a depressed information box with its text shown in white, whereas the cells in the current range have the background of their text (in their information boxes) shown in the highlight color.

To select a cell as the current one, simply click on it. Its information box will be depressed and its text shown in white. If you want a particular cell to be placed in the selected range, hold down the **Ctrl** key when clicking on the cell.

The background of the text will become green, but the text color will stay the same (black). To remove a cell from the range, do the same operation.

You can also select a specific range of cells by clicking on the first cell in the range, selecting the **Select Range** menu item, and clicking on the last cell in the range. All of the cells between, and including, these two will be highlighted.

NOTE When FRED works on the selected range, it will use the highlighted cells *as well as the current cell, even if the current cell's text is not highlighted*. This is an important point to remember.

You can also select all of the cells in a sequence by issuing the **Select All** menu item. Deselecting all of the cells is just as easy with the **Deselect All** menu item.

10.3.2 Arranging Cells

Cells do not have to stay in the same arrangement that they were added or inserted into the sequence—you can use the Clipboard to cut and copy cells from one part of the sequence and paste them in other parts. Note that only the information describing the cell, and **not** the image itself, will be posted to or retrieved from the Clipboard.

To remove a set of cells, select the cells and choose the **Cut** menu item. If you just want to copy the cells, then select **Copy** instead.

Pasting cells is done by selecting the cell that will precede the pasted information and selecting the **Paste** menu item. If no cell is currently selected, the pasted cells will be appended to the end of the sequence.

You can also paste the cells in reverse order that they were placed into the Clipboard by selecting the **Reverse Paste** menu item.

NOTE All cells that you cut or copy then later paste (using either paste command) keep their images' filenames. This is very important if you are going to process these images and overwrite the original version with the processed ones.

10.3.3 Loading Images Into ADPro

A handy feature is the ability to load a cell's image into ADPro. To do this, hold down the **Shift** key while double-clicking on a cell. That cell's image is loaded (using the UNIVERSAL) loader into ADPro. Note that no compositing settings are used—the image will replace whatever is currently in ADPro's buffer at the time.

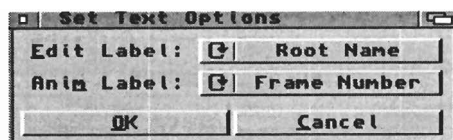


Figure 10.2: The Set Text Options control panel.

10.4 Cell Information

Various information about a cell can be displayed as well as set. This section describes how to do both.

10.4.1 Selecting the Cell Text

The **Text Options...** menu item under the **Settings** menu (available when no sequences are selected) will bring up the Set Text Options control panel (shown in Figure 10.2), from which you can change the text which appears below each stamp in a sequence as well as what text will be displayed during animation previews.

Both cycle gadgets (**Edit Label:** and **Anim Label:**) will let you select one of the following:

- | | |
|---------------------|--|
| Root Name | This is the root filename of the image, as it can be found on disk. This filename does not include any path information. |
| Frame Number | <p>A cell's frame number is a count of the number of "recordable units" which come before this cell. The first frame (first cell) in a sequence is frame one, although the first cell is cell zero. Each cell's length determines how many recordable units each frame occupies.</p> <p>For example, if a sequence contained just two cells each having a length of 1, then the frame number fields will show "00000001" and "00000002". If the first cell had a length of 60, then the cell's first frame number would still show "00000001" but the second frame number would show "00000061".</p> |
| Time | <p>This shows the exact time at which the image will be displayed in the animation. The computation of time is based upon the current animation frame rate.</p> <p>Time is expressed as minutes, seconds, and frames (or</p> |

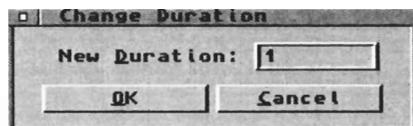


Figure 10.3: The Change Duration control panel.

Length

mm:ss:ff). A frame is a fraction of a second, whose duration is determined by the currently selected frame rate.

Size

This shows the number of recordable units in the animation that the current stamp will be displayed.

This is the width and height (in pixels) of the image. This information is available only if a stamp has been created for this image. “Stamp” is another word for the icon which FRED uses to represent each picture.

10.4.2 Setting the Duration

The **Duration...** menu item displays the Change Duration control panel (shown in Figure 10.3) from which you can change the number of recordable units the currently selected cells will occupy in the animation. All cells have a default time of 1.

The amount of time a given cell's frames will actually be displayed depends upon the currently specified frame rate. For example, if the current playback speed was 30 frames per second, an image having a time of 15 will be displayed for half a second when the animation is played.

This Change Duration value can also be entered in the Image Information control panel (shown in Figure 10.4). To bring up this panel, either double-click on the cell itself, or single-click on the cell and either select the **Information...** menu item (under the **Edit** menu) or press the **Return** key. In the Image Information control panel, you will see the filename of the cell, its currently defined length, the **Time In:** and **Time Out:** values, the **Frame In:** and **Frame Out:** values, and its **Width:** and **Height:**.

The **Time In:** and **Time Out:** values specify the display time (in seconds) for this cell. It takes into account the current **Animate Speed** (see Section 10.5) you have specified. For example, say that you have specified a speed of 15 frames per second and that the first three cells are of length 10, 5, and 20, respectively. The time in/out (frame in/out) values for these frames would be:

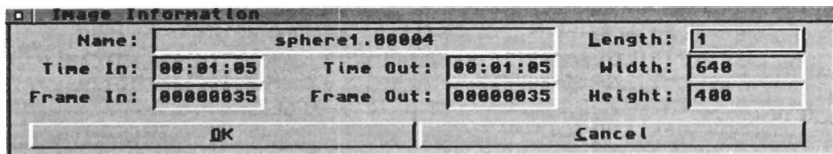


Figure 10.4: The Image Information control panel.

Frame		Time	
In	Out	In	Out
00000001	00000010	00:00:00	00:00:09
00000011	00000015	00:00:10	00:00:14
00000016	00000035	00:01:00	00:02:04

The **Frame In:** and **Frame Out:** values specify the first and last frame number described by this cell.

10.5 Animated Stamps

FRED allows you to preview your sequence of images in small, stamp-size images. You can specify the type of animated preview to produce, as well as the FPS (frames per second).

10.5.1 Creating Stamps

To create stamps for the currently selected cells, select the **Make Stamp...** menu item under the **Animation** menu. ADPro must be currently running for this operation to work. Each stamp will be placed in the same directory as the image it is making a stamp for, using a file extension of **".stp"** to denote that it is a stamp.

If you are running on a non-AGA machine (such as the A3000), the stamps will be rendered in 4 bit-planes (or 16 colors). If, however, you are running on an AGA-equipped machine (and the **SHALLOW** option isn't specified), then any generated stamps will be rendered in 8 bit-planes (or 256 colors).

10.5.2 Animating the Stamps

Once all of the stamps have been created, you can animate them by selecting various settings and operations in the **Animation** menu's **Playback**, **Range**,

and **Speed** submenus.

The **Animation** menu contains commands which allow you to set the frame rate (which affects how time is computed elsewhere), preview animations, create the stamps viewed in FRED, and cause generalized batch processing to take place. This menu is accessible only when a sequence window is selected.

You can choose to animate the entire sequence or the currently selected range by choosing either **All** or **Selected** in the **Range** submenu.

The speed of the animation playback can be defined by choosing the desired frame rate from the **Speed** submenu. Thirty FPS (frames per second) is the standard for NTSC video playback. Twelve FPS is useful for previewing animations which will be displayed on Commodore's CDTV. If FRED is running on a PAL machine, the options will be 25, 12.5, and 10 FPS.

The setting you choose here affects how time is calculated elsewhere in FRED. If you set 30 FPS, for example, FRED will increment the seconds counter (in the time code display) once each 30 frames. Setting the frame rate to 10 causes the seconds counter to increment every 10 frames.

To actually animate the frames using the current animation settings, select one of the three types of motion (**Playback**): **Once...**, **Continuous...**, and **Ping Pong...** **Once...** will play the frames only once. **Continuous...** will play them over and over again, continuously. The last frame is followed by the first frame. **Ping Pong...** is similar to **Continuous...**, except that the frame order reverses upon displaying the last frame.

A small window will appear, showing the animated frames. If you find that the frames are flickering, move the window to the bottom of the screen. This should reduce this problem. To stop the animation, click on the window's close gadget.

10.6 Sequence Processing

NOTE To process images with FRED and ADPro, you must either be knowledgeable in writing simple ARexx scripts or combining the pre-written ones. Consult Section 10.6.2 for more information on what needs to be included in a FRED ARexx script.

Your ARexx programs need not concern themselves with the details of loading the image to be worked upon, nor do they need to know how to pick the next image to be processed. These tasks are performed by FRED before calling the first ARexx program in the chain you specify.

FRED allows you to process either all of the images in a sequence or a select

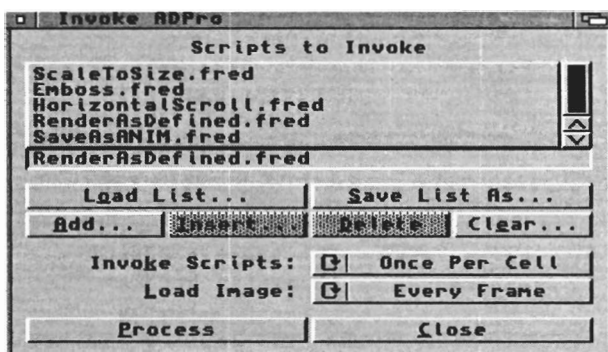


Figure 10.5: The Invoke ADPro control panel.

few. Note that as with the other commands that reference the currently selected cells, the currently selected cell (shown with its text in white) is part of the selected group, even if the background of its text is not highlighted (green).

Choose the images you want to process and select the **Process...** menu item. The FRED Invoke ADPro control panel (shown in Figure 10.5) will appear with the currently defined list of ARexx scripts. Notice that you can create a chain (list) of scripts (with the **Add...**, **Insert...**, and **Delete** buttons) that will be executed for each selected image in the current sequence. This allows you to assemble a complicated sequence of operations with smaller, more simpler, and more generalized scripts which you can reuse.

The buttons and cycle gadgets in this control panel are described below:

Add... A file requester will appear. Use it to enter the name of the ARexx script you would like to add. The selected ARexx program is added to the end of the chain.

Insert... This command allows you to add an ARexx script into the middle of the chain. First select the place in the chain where you would like to insert the script, then click on the **Insert...** button. A file requester will then appear in which you can enter the script name you would like to insert.

Delete This allows you to delete a script from the chain. First select the name of the script to be deleted, then click on the **Delete** button to remove the script from the chain.

Clear... This command allows you to start a new script list easier than having to individually delete each script in the list.

Invoke Scripts: This cycles between **Once Per Frame** and **Once Per Cell**, indicating when to invoke the list of scripts.

Load Image: This cycles between **Every Frame** and **Once Per Cell**, indicating when to load each image.

Once you are satisfied with the list of scripts and are ready to process the selected images, press the **Process** button. A meter window will appear, showing the progress of the operations.

If you want to return to the FRED control screen without performing the operations, select **Close**. These scripts will still appear in this list the next time you return to it.

10.6.1 Pre and Post Processing

In addition to the processing that will be performed on each selected frame in the sequence, you can also specify pre-processing and post-processing scripts. These scripts, along with the usage of ARexx's Clip List entries (attribute-value pairs), allow you to customize your Invoke ADPro scripts based on user input.

Pre-process scripts will be performed before any image data is loaded. Typically, you would use pre-scripts to ask the user to select certain options or set values used by the Invoke ADPro scripts.

Post-process scripts will be performed after all frames are processed. Typically, you would use these scripts to clean up the working environment, such as deleting temporary files or resetting or restoring Clip List entries.

These pre and post scripts are not specified directly by name, but are selected as a result of placing a particular script in the Invoke ADPro list of scripts. For example, if the "**SaveAsANIM.fred**" script (included with this package) was entered into the list, when you instruct FRED to execute the scripts it will try to find a script with a similar name, but with **".pre"** appended to it (in this case, "**SaveAsANIM.fred**"), to execute before any images are processed. After doing this pre-processing, FRED will execute the "**SaveAsANIM.fred**" script for each selected frame in the sequence. Once done, it will try to find and execute a script with a similar name, but this time with **".post"** appended to it.

NOTE When you select scripts to insert into the Invoke ADPro list, you are implying that the corresponding pre and post scripts should be executed as well. So, if you want a script to handle pre-processing for a given script, be sure to append **".pre"** to the name. For post-processing, be sure that you use **".post"**.

The order with which FRED will invoke the scripts is as follows:

1. All pre-process scripts for all defined Invoke ADPro scripts;
2. For each selected frame, execute the scripts listed in the Invoke ADPro list;
3. All post-process scripts for all defined Invoke ADPro scripts.

Included with the package are a set of “canned” scripts which you can use on your images. We hope that with these scripts you will find many more uses for FRED.

NOTE Some scripts might not be usable on your system configuration, either because it lacks the necessary hardware (display board, memory, etc.) or software (loader, saver, or operator modules or system software). The comment block (text at the beginning of each script) describes the script’s requirements. You can view this text by loading it into any text viewer or editor.

10.6.2 Contents of a FREDScript

There are a few issues you should understand when creating ARexx scripts for FRED (a.k.a. FREDScripts). This section will describe the structure of pre-process, main process, and post-process scripts.

Note that these descriptions are based on our own guidelines for creating scripts. However, you can structure you own scripts in whatever way you like.

General Rules and Information

You should always exit out of your scripts with a value (i.e., **EXIT** *n*). If *n* is non-zero, then FRED will immediately stop processing and return with an error message stating that a non-zero value was returned. If, however, if *n* is zero, then it will return to FRED’s screen without error.

Pre-Process Scripts

Pre-process scripts (scripts having an extension of “.fred.pre”) are executed *before* any of the selected images in the sequence have been processed. Most scripts use the pre-process script to retrieve information from the user to customize the settings used in the script, initialize some ARexx clip values, and create some temporary files for use during the processing.

The following information about the sequence is passed to the script:

- the total number of *selected* cells
- the total number of *selected* frames

Use the **PARSE** command to retrieve this information for use in your script, such as to ensure that there will be enough images to perform the operation. The included FREDScripts use the following line at the beginning of the pre-process scripts:

```
PARSE ARG NumberOfCells NumberOfFrames
```

Main Scripts

The main script (scripts having an extension of “**.fred**”) are executed for each image loaded into ADPro. The main processing is handled by the main script.

For each image, the following information about the sequence is passed to the main script:

- the current frame number (starting from 1).
- the current frame’s filename.
- the length (number of frames) of the current cell.
- a **Load Image:** indicator, which will either be 1 if the current image was loaded or 0 if it wasn’t.
- a first-frame-used indicator, which will be 1 if this is the first frame in the sequence to be processed.
- a first-frame-in-cell-used indicator, which will be 1 if this is the first frame in each selected cell to be processed.

Use the **PARSE** command to retrieve this information for use in your script, such as to display meaningful error messages and customize filenames used when saving an image to disk. You might want to use **PARSE ARG** instead to preserve the case of filenames. The included FREDScripts use the following line at the beginning of the main scripts:

```
PARSE ARG FrameNum FrameFName Length LoadFlag,  
FirstCallSeq FirstCallCell
```

Besides this line, the beginning of the main script usually has the commands to retrieve ARExx clip values.

What usually comes next in the script is the check for proper image data, be it raw or rendered data. A check for color or grayscale data, and possibly a specific number of colors, is also done at this point.

The end of the script usually has commands that update ARexx clip variables.

Post-Process Scripts

Post-process scripts (scripts having an extension of “**.fred.post**”) are executed *after* all selected images in a sequence have been processed through the main scripts.

No information about the sequence is passed to the post-process script.

Most post-process scripts handle the clean-up work for a script. Some examples include deleting temporary files used during the processing, resetting ARexx clips to their previous or default values, and properly closing all files used during the processing.

10.7 AnimOps Processing

FRED can also launch modular programs called “AnimOps” that work on sequence files. These AnimOps take one or more sequences (depending on its function) and operate on the images defined by the sequences.

The AnimOps bundled with this package are described in Chapter 11.

Chapter 11

AnimOps

The **AnimOps** menu is accessible only when the backdrop is selected and contains the name of each program found in the AnimOps directory (usually `ADP_FRED:AnimOps`).

AnimOps are programs which read sequence files and generate ARexx commands automatically controlling ADPro. Any programmer may create an AnimOp. Contact ASDG for more details.

The AnimOps supplied by ASDG with FRED are described in their own sections.

In this chapter, we will discuss the AnimOps supplied by ASDG with FRED. While all AnimOps should have certain elements in common (that is, if their authors have followed ASDG's design guide), details concerning AnimOps supplied by third parties will not be contained here.

11.1 Cinemorph

The Cinemorph (Drop Frame Processor) allows you to batch process many sequences of images recorded at either 24 or 30 frames-per-second (FPS). The program allows you to specify the loader and saver that is to be used during the processing so that you can change file formats during conversion. If you wish to send the files to something other than a disk file, you can specify additional options to control other modules, such as the ASDG Abekas Driver.

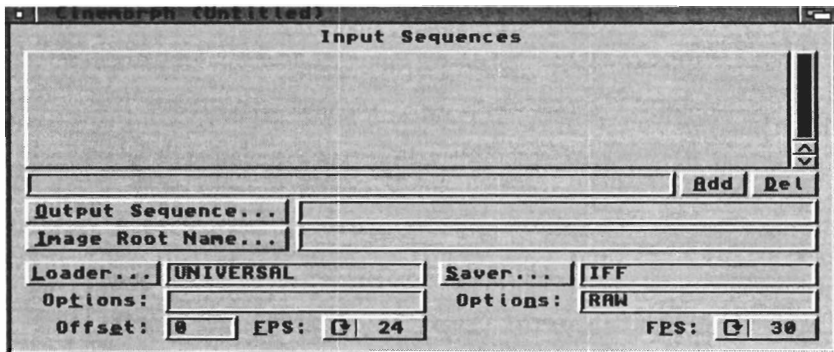


Figure 11.1: Cinemorph's main control panel.

11.1.1 Process Controls

Figure 11.1 shows Cinemorph's main control panel. The listview at the top of the window displays the list of sequence files to process. Each sequence entry in the list will be processed before moving to the next sequence, and each one has its own set of parameters for processing.

To add an entry into the list, select the **Add** button. This will bring up a file requester to allow you to specify the sequence file to add. These sequence files are FRED-style sequence files (with a ".seq" file extension). Only directories and files ending in ".seq" will be visible in the file requester.

To delete an entry from the list, you must first select the entry. Click on an entry in the list to select it. The entry will appear in the input field below the listview. Click on the **Del** button to delete the selected entry from the list.

To specify information for an entry, you must either add a new entry or click on an existing entry to change its information. When you click on an entry in the listview, the information for that entry is shown in the fields below the listview.

Note that the program will always keep a sequence entry selected, unless, of course, if there are no sequences in the list. Changes you make to a sequence's settings will be preserved, so if you switch from one entry to another, you will not lose the information you had in the previous entry.

If you want the program to recognize any changes you made to an input field value, you MUST press the Return key after typing in the value. Otherwise, it will be visually correct, but not internally.

11.1.2 Input Controls

Each sequence of images can be loaded different from the rest. This is done with the three controls in the lower left corner of the control panel.

To change the loader module used by ADPro to load in the sequence's images, either click on the **Loader...** button and select the loader from the file requester, or, if you know the name of the module, just type it into the input field to the right of this button. By default, the **ADPRO:Loaders2** directory is displayed. This entry must be specified.

To change the ARexx command options passed to the loader during the load operation, type them into the **Options:** input field located below the **Loader...** button. Not all loaders require arguments. This entry is optional.

To specify how many frames in the sequence to skip before processing begins, enter a value in the **Offset:** input field. This value is optional and defaults to 0.

11.1.3 Conversion Controls

The conversion between 24 and 30 frames-per-second is handled by the two **FPS:** cycle gadgets.

The left-side **FPS:** gadget specifies the frame rate that the sequence was created at. There are two choices. The setting of this gadget determines what type of conversion will take place.

The right-side **FPS:** gadget specifies the frame rate that the sequence is to be converted to.

If both gadgets are set to the same frame rate, then all that will be performed is a conversion to the output image format.

11.1.4 Output Controls

The **Output Sequence...** button, when selected, will bring up a file requester to allow the selection of an output sequence. This output sequence will be the sequence file for the new converted sequence. Optionally you can type in the complete path name of the output sequence file instead of using the file requester. This entry must be specified.

The **Image Root Name...** button, when selected, will bring up a file requester to allow the selection of an image base name. This base name will be used to write out each image in the converted sequence. Optionally you can

type in the complete path name of the image root name instead of using the file requester. This entry must be specified.

The **Saver...** button, when selected, will bring up a list of available savers to be used to save the images in the output sequence. By default, the **ADPRO:Savers2** directory is displayed. Optionally you may enter the name of the saver to be used. This entry must be specified.

The **Options:** input field under the **Saver...** button and input field is for specifying any additional saver options (i.e., saver-specific AReXX arguments) that may be needed for the saver you have chosen. This entry is optional.

11.2 Compositor

The Compositor allows you to easily create combinations of fades, wipes, and composites between two or more sequences of images using FRED. Users of previous versions of FRED may be familiar with the Compositor and Alpha.Compositor AnimOps that were included with the initial versions of FRED. This Compositor supersedes both of them, combining them into one AnimOp and including more options and controls.

The Compositor allows you to work on compositing “projects,” which can be created, stored, modified, and retrieved. Select the **Compositor** menu item from the **AnimOps** menu. A thin window bar will appear atop the FRED screen. All of the main controls are available from the AnimOp’s menu bar.

NOTE Many of the input fields in this AnimOp support the “exit help” feature. By placing the cursor within a specific input field and pressing the **Help** key on your keyboard, a panel will appear in the upper left corner of the screen with a description of that specific input field. Note that not all input fields have this capability.

11.2.1 Projects

To create a new compositing project, select the **New...** menu item. The small AnimOp window bar will expand to the Components control panel. This panel will be discussed in Section 11.2.2.

Projects can be created, stored on disk, opened, closed, and edited. All of the controls for projects are available in the **Project** menu.

To define the compositing of sequences, you must go through two steps:

1. specify the sequences to be composited

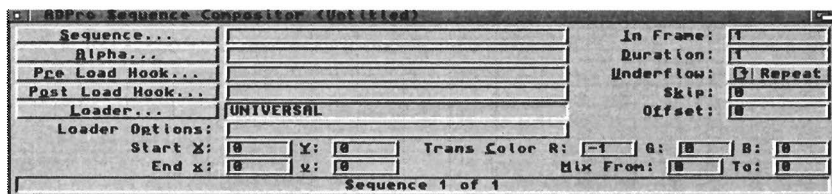


Figure 11.2: The Composer AnimOp's Components control panel.

2. specify the resulting sequence to create

The first step will be described in Section 11.2.2, while the second will be in Section 11.2.3.

11.2.2 Components

To display the Components control panel (as shown in Figure 11.2), select the **Define Components** menu subitem (from the **Mode** menu item). It is in this panel where you specify how one or more sequences will be composited together. When creating a new or loading a previously-saved project, the Composer opens in this mode.

You have complete control over when, where, and by how much each sequence's frames will mix with the frames of other sequences. Let's take a look at the contents of this panel.

The first thing you should spot is the **Sequence Indicator** text field at the bottom of the panel. It is here where you can identify one sequence from another. You can flip from sequence definition to definition by using the **Next** and **Previous** menu items (in the **Edit** menu). This field will display which sequence (of the total number of sequences) you are currently editing.

Sequences are numbered starting from 1. The sequence defined as Sequence 1 can be thought of as being at the bottom (position-wise) of the sequences that will be defined. The other menu items in the **Edit** menu allow you to add, delete, copy, paste, and erase sequences from this "stack" of sequences.

The sequence file is described in the **Sequence...** input field. You can either enter a new filename into the field or press the button to select one from the file requester. Only files ending in **.seq** will be visible. When saving sequence files out of FRED or any other program which outputs sequence files, you must ensure that this filename extension exists for FRED and these other programs to recognize it as a valid sequence file.

You can specify the transparency of the images in the sequence with another

sequence of grayscale images (of the same dimension as the images in the sequence to composite) by entering the sequence's filename in the **Alpha...** input field. This alpha channel sequence file should be of the same length as the primary sequence itself.

The **Pre Load Hook...** and **Post Load Hook...** input fields can contain the filenames of ARexx scripts which will be executed before and after loading each frame of the sequence. This feature gives you a lot of flexibility in manipulating the images in your sequence, without having to modify the originals—you can use the same master sequence with different hooks to produce a potentially drastic difference set of images.

The **Loader...** and **Loader Options:** text and input fields allow you to specify the type of images that are contained within the current sequence, and any loader-specific arguments which might be required. By default, the UNIVERSAL loader is used.

The **Start X:** and **Y:** values control the initial position of the upper left hand corner of the images in this sequence, relative to the currently defined “backdrop” image. The **End X:** and **Y:** values specify the final position of this same corner for the last frame to be processed in this sequence.

The “backdrop” image for a given frame number will not always be the image defined in the bottommost sequence (Sequence 1). Depending upon the **In Frame:** number (described later in this section), a “higher” sequence might actually constitute the backdrop.

The transparent color in the current sequence's images can be defined with the **Trans Color R:**, **G:**, and **B:** values. If **R:** is set to -1, then there will be no transparent color (the image will be opaque). If each of these values are set to a number between 0 and 255, then any pixel in the image that is this color will be transparent, and that the next lowest opaque pixel (from lower sequences) will “show through.” Note that if a current frame is the backdrop image, then these transparency values are ignored.

The amount of mixing that will be done, from the first frame to process to the last, is defined in the **Mix From:** and **To:** input fields. These values can range between 0 and 100.

The actual frames (in this sequence) that will be composited is controlled by the values in the input fields and cycle gadget located at the right of the panel.

In Frame: describes at which frame number (sequences are described as starting with a frame number of 1) in the resulting sequence this current sequence's frames will start to be composited. The first frame (in this sequence) to process is specified in the **Offset:** input field. The number of frames to process is specified in the **Duration:** field. If you want to process every other frame, or every third or fourth frame, then you can change the value in the **Skip:**



Figure 11.3: The Compositor AnimOp's Results control panel.

input field to tell the AnimOp how many frames to skip after finishing with a particular frame.

If the resulting sequence “fills up with images” (the total number of frames requested has already been reached) but you still have frames in the current sequence to process, then those frames will not affect the final sequence at all. If, however, you use up all of the selected frames in the sequence before the resulting sequence finishes, then the setting of the **Underflow:** cycle gadget will define what to do with the current sequence.

If set to **Repeat**, then the last frame that was processed will be used for the rest of the resulting sequence. If set to **Stop**, then no more frames from this sequence will take part in the compositing.

11.2.3 Results

Once you have defined the sequences that will be composited together, you would then define the resulting sequence that will be generated. Switch to the Results mode by selecting the **Define Results** menu subitem. The Compositor's Results control panel (as shown in Figure 11.3) will appear.

The name of the resulting sequence must be defined in the **Output Sequence...** input field. The composited frames will be saved (appended by the sequence's current frame number) with a base (root) name, as defined in the **Image Root Name...** input field. If you want the frames to be saved as **Work:Images/vacation.00001**, **Work:Images/vacation.00002**, and so forth, then enter **Work:Images/vacation** as the root name.

If you require special processing of the image before or after it is saved to disk, then specify the ARexx scripts in the **Pre Save Hook...** and **Post Save Hook...** fields, respectively.

If you require special processing of the image before or after any other processing is performed, then specify the ARexx scripts in the **Pre Script Hook...** and **Post Script Hook...** fields.

The format in which these images will be saved must be specified in the

Saver... text field. Any saver-specific options that need to be included should be entered into the **Saver Options:** input field.

Lastly, the total number of frames in the resulting sequence is defined by the value in the **Total Frames:** input field, the number of frames in the resulting sequence that will be skipped (these frames will only contain the current background image) in **Skip Frames:**, and the number of frames to process in the resulting sequence (starting after the skipped frames) in **Process Frames:**.

NOTE If you have previously generated a sequence of frames but find out that one or more frames was not generated the way you intended, you can regenerate a range of frames by changing the **Skip Frames:** and **Process Frames:** values to tell the Compositor which frames should be remade. You should only change these two values—and **not** the values in the component sequences. This will ensure that the frames you want to regenerate use the same position and mix values as the original frames.

Once you are satisfied with the project, you may want to save it before actually initiating the compositing. To begin the process, select the **Accept** menu item (from the **Project** menu). To exit this AnimOp, select **Quit Compositor**.

11.3 Time_Stretch

The Time_Stretch AnimOp (also referred to as the “time stretcher”) allows you to increase or decrease the number of frames in a given sequence by merging or interpolating image data. For example, suppose you rendered an animation at 10 frames per second. Playing this animation back at 30 FPS either results in triple speed (if you display each frame for $\frac{1}{30}$ of a second) or jerky motion (if you display each frame for $\frac{3}{30}$ of a second).

Time_Stretch can create additional frames containing in-between multiple exposures easing from one original frame to the next. If the results are played back at 30 FPS, the animation will look smoother than the original (though, of course, not as good as if you had actually rendered it at 30 FPS in the first place).

On the other side of the coin, Time_Stretch can shorten time. Suppose you have a 30 FPS animation but you know your display device cannot possibly display 30 FPS (but can achieve 15 FPS). You could correct for this by simply leaving out every other frame but this will make you animation look choppy.

You can tell Time_Stretch to change 30 FPS into 15 FPS. It will produce a sequence containing the information of two 30 FPS frames in every 15 FPS

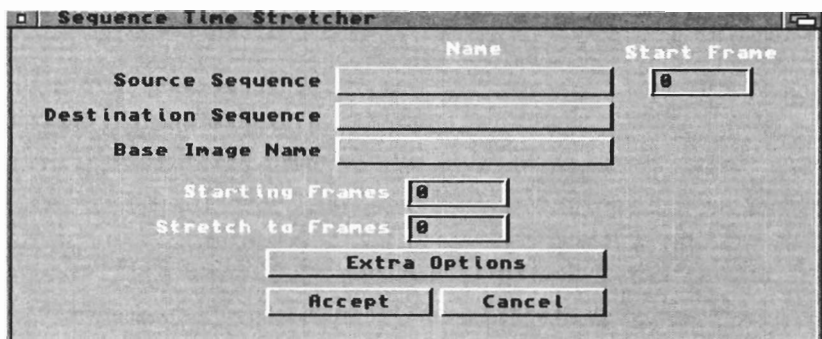


Figure 11.4: The Time.Stretch AnimOp's main control panel.

frame. The results look much more fluid than simply dropping frames.

11.3.1 Main Control Panel

The main control panel (shown in Figure 11.4) allows you to select the sequence to stretch, as well as define to how many frames it should be stretched.

Source Sequence This is the sequence which you would like to time stretch. To select a sequence, select the **Source Sequence** button and a file requester will appear for you to select the sequence name.

Start Frame This is the frame offset at which Time.Stretch will begin. This allows you to stretch a portion of a sequence, affecting only the frames including and after the offset number. The offset must be less than the total number of frames in the sequence.

Destination Sequence This is the resulting sequence produced by Time.Stretch. To specify a filename, select the **Destination Sequence** button and a file requester will appear. If you are recording directly to a single frame device, you can leave this field blank.

Base Image Name This is the base name of the images which are created in the destination sequence. This allows you to specify a name for the images that were created by the compositor. If the base name was **pic**, all of the images would have the name **pic** with the frame number as a suffix. For example, the image in frame number 30 of the sequence created would be named **pic.00030**.

Starting Frames This is the number of frames in the source sequence which you would like to stretch. This number may not exceed the length of the source sequence. A beginning frame number other than zero may be

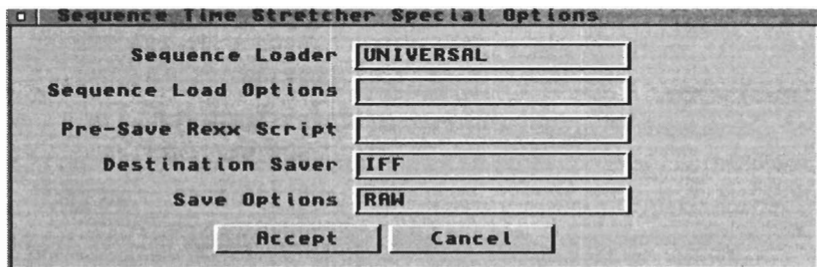


Figure 11.5: The Time_Stretch AnimOp's Extra Options control panel.

specified by entering a number in the **Start Frame** input field. This will allow you to take a portion of a sequence instead of the entire sequence.

Stretch To Frames This is the number of frames you would like Time_Stretch to create from the source sequence. If you specify more frames than are in the original sequence, the new sequence will become longer. If the sequence has more frames than the number of frames specified, the sequence will be compressed.

11.3.2 Extra Options Control Panel

This control panel (shown in Figure 11.5) allows you to customize the ARexx commands that FRED will send to ADPro. For instance, if you know the background sequence contains only images in the IFF format, then you might specify IFF as the background loader rather than UNIVERSAL, an ADPro-only loader, which will speed things up a little.

Background Loader This specifies the loader that ADPro will use when loading the images for the background sequence. Any valid loader name in ADPro may be used.

Background Load Options This enables you to specify the loader options ADPro will use. To specify an option refer to the ARexx section of the ADPro manual. The loader options are the same as the ARexx options for each of the different loaders.

Foreground Loader This specifies the loader that ADPro will use when loading the images for the foreground sequence. Note that the foreground and background loaders do not have to be the same.

Foreground Load Options This enables you to specify the loader options ADPro will use. To specify an option refer to the ARexx section of the ADPro manual. The loader options are the same as the ARexx options for each of the different loaders.

Pre-Save ARexx Script You can specify the name of an ARexx program here which will be called immediately after the compositing functions are completed but prior to saving the results. You can tie in single frame recorders here, for example, by causing the image to be saved to a display device and then issuing the ARexx command to record.

Destination Saver This is the file format you would like to use for saving composited images. You may specify any of ADPro's savers. Blank out this string if you don't want the AnimOp to directly cause a save to take place.

Save Options This allows you to specify the options that the saver will use. These arguments may be found in the ARexx section of the ADPro manual or reference section of the ADPro manual.

Chapter 12

ARexx Interface

By using the powerful scripting language, ARexx, ADPro becomes an enormously flexible automated image processing system. Nearly every aspect of ADPro can be controlled from ARexx using simple English-like commands. This chapter describes the usage of ADPro from ARexx. For specific ARexx command specifications, please refer to the various sections in Chapter 13.

NOTE You must have ARexx in order to use the facilities described in this chapter and in Chapter 13. Specifically, if ADPro cannot locate the `rexsyslib.library` file (in the `LIBS:` directory), then it will not respond to any ARexx style requests.

12.1 General Information

This section describes how to communicate with ADPro via ARexx and how to receive results from ARexx commands. The general rules for ARexx programming, in relation to ADPro, are also discussed.

12.1.1 Addressing ADPro

In order to access ADPro or any of its Loaders, Saver, or Operators from ARexx, you must tell ARexx how to make contact with ADPro. Specifically, you must tell ARexx the name of the message port which ADPro creates for the purpose of ARexx interaction. The name of this port is:

ADPro

You can make the connection from ARexx to ADPro using the following ARexx command:

```
ADDRESS "ADPro"
```

ARexx will generate an error condition should ADPro not be found.

NOTE The ADPro ARexx port name is case specific. It must be specified exactly as shown above, otherwise interaction will not be possible.

12.1.2 Getting Results of Commands

After every command directed at ADPro from ARexx is executed, ADPro will fill in the standard ARexx result variable, **RC**, with a return value. If this value is 0, then the previous command executed without error. If the value of **RC** is not zero, then it contains an error severity indicator.

In order to receive additional error indications or to receive non-error status information back from ADPro, you must request additional status information from ARexx using the following command:

OPTIONS RESULTS

This command must be one of the first executable instructions in your ARexx program.

If an **OPTIONS RESULTS** command has been issued, then ADPro will load additional information into the ARexx variable **ADPRO_RESULT**. **ADPRO_RESULT** can be consulted whenever **RC** is non-zero to gain additional information about the nature of an error.

When **RC** is returned with a 0 value (indicating no error occurred), **ADPRO_RESULT** will be set to command specific information. Note that this means you must check **RC** before depending on the data in **ADPRO_RESULT** since **ADPRO_RESULT** may contain either additional error information or the status information you requested.

The **ADPRO_RESULT** variable is available only to true ARexx programs. If you are sending ARexx style commands to ADPro from a non-ARexx program, then results which normally would be passed back in **ADPRO_RESULT** will instead be passed back in the **rm.Result2** field in the ARexx message structure.

The ARexx command specifications (in Chapter 13) usually list how each command affects the `RC` and `ADPRO_RESULT` variables. If either one or both is missing (or if the variable is labelled “unchanged”), then that particular command doesn’t change the unlisted variable’s value. If it is labelled “undefined”, then the value may or may not have changed. This label means you should not depend on the value of this variable.

12.1.3 Invoking Commands

Most of the ARexx commands return information in `ADPRO_RESULT`, especially when an error has occurred. The release of ADPro to the public marks the first major distribution of modules which use a slightly different form of ARexx command invocation and error detection. This technique of interfacing with a program (or module) through ARexx should make the task of writing ARexx programs more flexible.

Not all of the modules (loaders, savers, operators) use this system to handle command parsing. Most of the time, you will be able to tell which method is being used just by the freedom with which you can specify module-specific options or arguments. In the old style, these arguments usually had to be given in a defined order. In the new style, these same arguments can be listed in any order (within specific constraints).

For example, the ARexx interface for the Roll operator (an old-style interface) is:

```
OPERATOR "ROLL" direction pixels or
OPERATOR "ROLL" direction pixels "WRAP"
```

Notice how the *direction* value must come immediately after the `OPERATOR "ROLL"` sequence and before the *pixels* value. Under the new-style interface, that same module could be controlled with the following syntax:

```
OPERATOR "ROLL" arguments
```

where *arguments* can be zero or more of the following (listed in any order):

```
DIRECTION direction
```

```
PIXELS pixels
```

```
WRAP
```

If one of the above *arguments* were missing, then the last value for that particular *argument* will be used. Also, being able to list these arguments in any order frees you from having to remember which argument should come before

another.

Another advantage that these modules have over their old-style counterparts is the improved error handling. Say, for instance, you forgot to include the required **PIXELS** argument in your ARexx command. This would be sensed, giving you an error code in **RC** and a descriptive message in **ADPRO_RESULT**, such as, "**No arg found in same class as: PIXELS**". From this information, you would be able to determine which arguments need to be specified. If more than one required argument is missing, however, only one of the missing required arguments will be described in **ADPRO_RESULT**.

In addition to checking for required arguments, this new system also confirms that given values are within the valid range of values (as specified by the module itself). For example, say you typed in **PIXELS 500** but the valid range was only 0 to 250, the module would (again) tell you what is required in **ADPRO_RESULT**.

12.1.4 Understanding the General Rules

Three general rules apply to all ADPro ARexx commands. These are:

1. All commands set **RC** and **ADPRO_RESULT** as indicated above.
2. Most commands which change some internal state in ADPro return the old value in **ADPRO_RESULT** if no error occurred. If a command which requires additional arguments to change an internal state is executed without the required arguments, the old value of the state is returned and is not changed. Therefore, a command which changes some state can also be used to interrogate the present value of the state without changing it.
3. All commands are case-insensitive. However, arguments may be case sensitive.

To create ARexx scripts for ADPro, please note the following:

- Always begin your script with a comment block—all ARexx scripts **require** this. Information which is recommended for this block of text include the name of the script and a description of the script's function. You might also want to place a version string of the form:

\$VER: *script_name version (dd.mm.yy)*

- Place the **ADDRESS "ADPro"** and **OPTIONS RESULTS** commands near the beginning of your script so that you can receive results from the commands.

Key Sequence	F1	...	F9	F10
F key only	F1.adpro	...	F9.adpro	F0.adpro
Shift + F key	SF1.adpro	...	SF9.adpro	SF0.adpro
Alt + F key	LF1.adpro	...	LF9.adpro	LF0.adpro
Amiga + F key	AF1.adpro	...	AF9.adpro	AF0.adpro

Table 12.1: Naming conventions for function key macros.

- Always try to check for the success of every command. If a script isn't working the way you intend, having error checking code in your script will help you determine more easily the command causing the error.
- Always try to include comments within your script. You might know what the script does now, but in the future you might come across the same script and have no or little idea what it does.
- When you have to exit the script with the **EXIT** command, be sure to exit with a return value. Use **EXIT 0** for successful or no error exits, and **EXIT 10** (or whatever other non-zero value) for errors or operations stopped by the user. Even though the program calling the ARexx script might not pay attention to these error codes, other programs might. It is always good to know whether a script ended successfully or not.

The best way of learning how to write an ARexx script for ADPro is to examine one of the pre-written scripts bundled with this package. They have been designed with the above rules.

12.2 Launching ARexx Scripts

ADPro has always been known for its batch processing capabilities. These come about because of its extensive ARexx support. This section describes the various ways of invoking ARexx scripts from ADPro.

12.2.1 Keyboard Macros

ADPro can launch more than 100 distinct but specially named ARexx programs, each one linked to a particular key or sequence of keys on the keyboard. These programs must reside in your **REXX:** directory (or in the directory specified by the **REXXDIR** icon Tool Type or **RD** command line argument) and must be specially named according to the conventions listed in Tables 12.1 and 12.2.

Key Sequence	KeyPad1	...	KeyPad9	KeyPad0
KeyPad key only	N1.adpro	...	N9.adpro	N0.adpro
Shift + KeyPad key	SN1.adpro	...	SN9.adpro	SN0.adpro
Alt + KeyPad key	LN1.adpro	...	LN9.adpro	LN0.adpro
Amiga + KeyPad key	AN1.adpro	...	AN9.adpro	AN0.adpro

Table 12.2: Naming conventions for numeric keypad macros.

If more than one of the **Ctrl**, **Shift**, **Alt**, or **Amiga** keys are depressed at the same time as a function key, ADPro will *not* attempt to launch an ARexx program.

If one of the key combinations (described in Tables 12.1 and 12.2) is depressed for which there is no corresponding ARexx program, you will be told.

Also, you can use almost all the keys on the keyboard (other than the function and keypad keys) to launch scripts. These scripts are prepended with a **K**. For example, if you want to bind a key to the **\$** symbol, you would place a script called **K\$.adpro** into your **REXX:** (or **REXXDIR/RD**-defined) directory.

NOTE The only qualifiers which will work on the **K** scripts are the **Amiga** key (where scripts will be called **AKkey_sequence.adpro**) and the **Ctrl** key (**CKkey_sequence.adpro**).

Also, if you use the **Return** key, it will be changed to **M** (i.e., **CKM.adpro**); note that **Return** is represented as **Ctrl + M**, that's why it tries to find **CKM.adpro**. Pressing the **Space bar** will change the space character into an underscore (i.e., **K_.adpro**).

Pre-Written Scripts

We provide an ARexx program attached to the **F10** function key which allows other ARexx programs to be launched by name. This program, called **F0.adpro**, will ask you to specify an arbitrary ARexx program to execute. This allows you to use ARexx programs which do not conform to the naming conventions listed above. The **F0.adpro** program comes on the ADPro distribution disk. You can install it into your **REXX:** (or **REXXDIR/RD**-defined) directory by running the installation script and choosing the option that installs the sample ARexx programs.

As a bonus, we have created ARexx scripts called **N0.adpro** through **N9.adpro**, **SN0.adpro** through **SN9.adpro**, **LoadSwapBuff.adpro**, and **SaveSwapBuff.adpro** to "simulate" 10 swap buffers in your **T:** assigned directory. install the ADPro scripts in your in any other directory).

For example, to save the raw image data currently in ADPro to swap buffer 4, press **Shift + KeyPad4**. The script will save the image under the filename "T:ADProTemp.4", where the extension is the number of the swap buffer. To load an image from another buffer, say buffer 0, simply press **KeyPad0**.

NOTE You can bind any script to these keys just by changing the name to something that ADPro will recognize. We have included these pre-written scripts for your convenience.

There are other pre-written scripts which we have included. Consult the ReadMe file for more information.

12.2.2 Pseudo Modules

ARexx scripts can also be launched as if they were Loader, Saver, or Operator modules just by placing them in any of the ADPro subdirectories (named "Loaders2", "Savers2", and "Operators2"). The ADPro program will recognize that these are ARexx scripts and launch them accordingly.

This will allow you to use your existing ADPro ARexx scripts from within the interface as if it were another module.

NOTE Although you can manually invoke these pseudo-loader/saver/operator modules, do **NOT** try to invoke them from an ARexx script—they will not work. Although you can call any other module from these scripts, the only restriction is that you **CANNOT** call the script itself.

12.2.3 User Commands

The "Commands2" directory is also scanned for ARexx scripts, although they are displayed within ADPro's List and Button Interfaces. They differ from the other types of scripts in that they are easily accessible from the main window and do not have to be named a certain way.

In addition to all of the pre-written scripts for ADPro, you can install your own scripts within this directory.

What is useful with User Commands is that your favorite scripts can be placed within the menus (under the **User** menu), just like the loaders, saver, and operators.

NOTE This directory is only read at the time ADPro is started, so changes to this directory while ADPro is running will not be apparent until

the next time ADPro is started.

12.2.4 Execute ARexx Script

If the previous methods of invoking scripts weren't enough, you can invoke any ARexx script (of any name, located in any directory) with the **Execute ARexx Script...** menu item under the **User** menu.

Selecting the **Select ARexx Script...** button display a file requester, from which you can choose any script from any directory to invoke.

12.3 ARexx Commands Cross Reference

For a complete listing of all ARexx commands available in ADPro, please refer to the various sections in Chapter 13. These ARexx commands are grouped by subject matter—main window functions, loaders, savers, and operators. Commands which do not belong to any of these groups are listed in Section 13.7 in Chapter 13.

Listed below is a complete cross reference of all ARexx commands available in ADPro, along with page references to their descriptions. The commands are listed in alphabetical order.

Command	Description	Page
ABS_SCALE	Scale the image to absolute dimensions.	446
ADPRO_DISPLAY	Display the rendered image.	456
ADPRO_EXIT	Exit the program.	350
ADPRO_TO_BACK	Move the ADPro screen behind all others.	352
ADPRO_TO_FRONT	Move the ADPro screen in front of all others.	352
ADPRO_UNDISPLAY	Remove the display of the rendered image.	456
AVAIL_MODES_ONLY	Set/get availability of all render modes.	457
BLUE	Set/get the blue content of the rendered image.	354
BRIGHTNESS	Set/get the brightness of the rendered image.	355
CLEAR_RAW	Remove the raw image data.	457
CLEAR_RENDERED	Remove the rendered image data.	457
CLOSE_RENDER_SCREEN	Close the render screen.	458
CONTRAST	Set/get the contrast of the rendered image.	355
CURRENT_MEM_SIZE	Get the current size of the image buffer.	350
DISPLAYMESSAGE	Display a message in the title bar.	372
DITHER	Set/get the dither method to use.	460
EXECUTE	Create rendered data from raw data.	455
GAMMA	Set/get the gamma correction for the image.	355
GCR	Set/get Gray Component Replacement value.	423
GETDIR	Ask the user to select a directory.	369
GETFILE	Ask the user to select a file.	367
GETFILES	Ask the user to select a list of files.	368
GETLIST	Get various program (internal) lists.	371
GETNUMBER	Ask the user to enter a number.	366
GET_SCREEN_MODE	Ask the user to select a screen mode.	370
GETSTRING	Ask the user to enter a string.	365
GREEN	Set/get the green content of the image.	354
IMAGE_TYPE	Get the type of image data available.	461
INK_CORRECTION	Set/get ink correction value.	424
INTERFACE	Set/get ADPro's interface type.	352
INTERFACE_SIZE	Set/get ADPro's interface size.	353
LAST_LOADED_IMAGE	Get the filename of the last loaded image.	364
LAST_SAVED_IMAGE	Get the filename of the last saved image.	364
LFORMAT	Set/get the loader format.	375
LISTVIEW	Choose from a list of items.	371
LOAD	Load an image into ADPro.	375
LOAD_DEFAULTS	Load ADPro's settings from a file.	373
LOAD_GUI	Launch the selected loader.	376
LOAD_TYPE	Set/get the status of the Compositing button.	379
LOADER	Execute a loader module.	376

Command	Description	Page
MAGENTA_CORRECTION	Set/get magenta correction value.	423
MPPOKE	Set/get multiple palette register values.	361
OBTAIN_ADPRO	Obtain exclusive lock on ADPro.	374
OFORMAT	Set/get the operator module.	427
OKAY1	Display a message with one button.	372
OKAY2	Display a message with two buttons.	373
OKAYN	Display a message with user-definable buttons.	373
OPERATE	Perform the currently selected operator.	427
OPERATE_GUI	Launch the selected operator.	428
OPERATOR	Execute an operator module.	427
ORIENTATION	Set/get the load orientation.	379
PAUSE	Cause a delay in the execution of the script.	365
PCONTRAST	Define whether colors 0 and 1 should contrast.	359
PCT_SCALE	Scale the image by percentages.	446
PGETWB	Load the first four Workbench colors.	362
PIXEL_ASPECT	Get the image's pixel aspect.	462
PIXEL_RESOLUTION	Get the image's pixel resolution.	462
PIXELPEEK	Get the colors from a rectangular area of pixels.	361
PIXELPOKE	Set the colors for a rectangular area of pixels.	361
PLOAD	Load a palette file.	362
POFFSET	Specify a register as offset color zero.	358
PPOKE	Set/get the value in a color register.	360
PSAVE	Save the contents of the palette to a file.	363
PSORT	Set/get the sort ordering of the palette.	360
PSTATUS	Set/get the palette lock mode.	357
PTOTAL	Set/get the total number of colors.	357
PUSED	Set/get the number of colors to actually use.	358
PWIDTH	Set/get the accuracy of the palette.	356
RDEPTH	Set/get the status of render depth usage.	456
RED	Set/get the red content of the image.	353
RELEASE_ADPRO	Release exclusive lock on ADPro.	374
RENDER_TYPE	Set/get the rendering mode.	455
RMODE	Set/get the status of render mode usage.	456
SAVE	Save the image currently in ADPro.	397
SAVE_DEFAULTS	Save ADPro's settings to a file.	374
SAVE_GUI	Launch the selected saver.	398
SAVER	Execute a saver module.	398
SCREEN_TO_FRONT	Move a public screen in front of all others.	352
SCREEN_TYPE	Set/get the type of screen to use.	458
SEPARATE	Perform a color separation.	425

Command	Description	Page
SEPARATION_DEPTH	Set/get color separation depth.	424
SEPARATION_MAP	Set/get color map type.	423
SEPARATION_TYPE	Set/get color separation type.	424
SERIAL_NUMBER	Get the program's serial number.	350
SET_ADPRO_MODE	Set the non-public screen to open ADProon.	351
SET_ADPRO_PUBLIC	Set the public screen to open ADProon.	351
SET_RENDER_MODE	Set the render screen.	459
SETPALINFO	Set/get palette color information.	359
SFORMAT	Set/get the saver format.	397
UCR	Set/get Under Color Removal setting.	422
USE_SEPARATION- _DEFAULTS	Set UCR/GCR/ink correction to default values.	424
VERSION	Get the program's version number	350
XSIZE	Get the width of the image.	461
YELLOW_CORRECTION	Set/get yellow correction value.	423
YSIZE	Get the height of the image.	461

12.4 ADProScripts

This section gives a brief description of all the ARexx scripts for ADPro (ADProScripts) included with the package. More information can be found within the scripts themselves. Use a text editor to view and edit the files.

BustANIM.adpro

This script takes an ANIM and separates (or busts) it into individual frames. The original ANIM is not altered in any way. You can even specify the starting and ending frames to “bust”.

CheckDPI.adpro

This script checks for the existence of a “DPI” chunk in a file. It uses the ZapDPI utility.

ConvertANIM.adpro

This script takes an ANIM and converts it between ANIM-5 and ANIM-8 formats, creating a new ANIM file.

CropANIM.adpro

This script takes an ANIM and creates another ANIM whose frames are the cropped out areas of the original ANIM.

F0.adpro

This script (bound to the **F10** key) actually allows you to launch any other script. We suggest you use the new **Execute ARexx Script...** command under the **User** menu instead.

FlipANIM.adpro

This script takes an ANIM and creates another ANIM with its frames flipped either vertically or horizontally (user-specified).

GETFILES_Examples.adpro

This script is an example of using the **GETFILES** ARexx command. We suggest you try to use the **GetAListOfFiles** FREDScript instead.

GetFileVersion.adpro

This script displays the version string (version number and date) of a selected file. It is a general purpose version number displayer.

GetModeID.adpro

This script displays the mode identifier (mode id) for a given screen mode. A screen mode requester is displayed for the selection of a mode. ARexx programmers will find this script helpful in creating custom scripts which set the render mode.

GetModuleVersion.adpro

This script displays the version string (version number and date) of a selected ADPro-type module (loaders, savers, operators, etc.). This script is a more specific version of the **GetFileVersion.adpro** script.

JoinANIMs.adpro

This script takes two or more ANIMs and joins them together (one concatenated after the other) into one ANIM file.

Joinz.adpro

This script is a front-end to the Amiga Joinz utility, which is used to rejoin the chunks of previously-split files.

Last_Loaded.adpro

This script displays the filename of the last loaded image.

LoadSwapBuff.adpro

This script should not be executed by itself—it is used by the **N0.adpro** through **N9.adpro** scripts to load a specific swap buffer.

Locate-ADPro.adpro

This script tries to find and start the ADPro program. It searches for in the **ADPRO:** directory. Note that the program is started with the **MAXMEM** Shell argument set to 6,000,000 bytes.

Make-HAME.adpro

This script creates the data suitable for display on a HAM-E device.

N0.adpro - N9.adpro

These scripts (bound to the keypad's 0 through 9 keys, respectively) are used to load a specific swap buffer's contents.

PingPongANIM.adpro

This script takes an ANIM and creates another ANIM with the original ANIM's frames ping-ponged.

ReDPI.adpro

This script adds a “DPI” chunk to a file using the ZapDPI utility.

RemapANIM.adpro

This script takes an ANIM and, given a palette file, rerenders the ANIM using that palette, creating a new ANIM.

ReRenderANIM.adpro

This script takes an ANIM and rerenders it in a different screen mode, number of colors, and dither from the original, creating a new ANIM.

ReverseANIM.adpro

This script takes an ANIM and creates another ANIM with the frames of the original saved in reverse order.

SaveSwapBuff.adpro

This script should not be executed by itself—it is used by the **SN0.adpro** through **SN9.adpro** scripts to save to a specific swap buffer.

ScaleANIM.adpro

This script takes an ANIM and creates another ANIM whose frames are scaled versions of the originals.

ScaleToFillArea.adpro

This script scales the image to best fill a user-definable area but preserving the original image aspect (proportions).

ScaleToPixelAspect.adpro

This script scales the current image in the buffer to a specific pixel aspect. This is very useful when transferring images between platforms whose displays

have different pixel aspects.

SN0.adpro - SN9.adpro

These scripts (bound to the keypad's 0 through 9 keys, respectively) are used to save the current raw image data to a specific swap buffer. Hold down the **Shift** key while pressing one of the keypad keys.

SplitANIM.adpro

This script takes an ANIM and splits it into two separate ANIMs.

Splitz.adpro

This script is a front-end to the Amiga Splitz utility, which is used to split a file into several chunks.

ZapDPI.adpro

This script removes the "DPI" chunk from a file, using the ZapDPI utility.

12.5 FREDScripts

This section gives a brief description of all the ARexx scripts for FRED (FREDScripts) included with the package. More information can be found within the scripts themselves. Use a text editor to view and edit the files.

12.5.1 FREDOperators

AdjustBalancing.fred

This script allows you to adjust the color balancing settings on your images. You can either modify them in batch mode (globally) or interactively.

Antique.fred

This script applies the Antique operator on your images.

ApplyMap.fred

This script filters the raw image data through the current color balancing settings. You can preview your balancing changes before rendering by invoking the **Redisplay.fred** script.

Blur.fred

This script performs a blur operation or tests how many pixels would be affected by the current settings on your images.

BroadcastLimit.fred

This script performs a broadcast limit operation or tests the current settings on your images.

Colorize.fred

This script applies the colorize operator on your images.

ColorToGray.fred

This script converts all color images to raw grayscale data. Grayscale images are skipped.

Convolve.fred

This script applies a pre-saved convolution matrix on your images.

CropToSize.fred

This script crops out the same rectangular area within your images.

DefinePxlAspect.fred

This script sets the pixel aspect and DPI of a sequence of image to the same values.

DeInterlace.fred

This script separates an image into two fields stacked one atop the other. This is useful when framegrabbing images as it can remove unwanted results where rapidly moving elements are present in the image.

DisplacePixel.fred

This script scatters the pixels in your images across a specified number of frames. You can explode, implode, explode then implode, or implode then explode the sequence of images.

DynamicRange.fred

This script applies a dynamic range operation on your images. Video operators could use this script to prevent NTSC video smear.

Emboss.fred

This script applies an emboss effect on your images.

Fresco.fred

This script applies the Fresco operation on your images.

GradualCollapse.fred

This script applies the Collapse operation on your images. You can specify the starting and ending Collapse settings.

GradualCrop.fred

This script crops a rectangular area out of a sequence of images. You can specify the starting and ending crop settings.

GradualMosaic.fred

This script applies the Mosaic operation on your images. You can specify the starting and ending Mosaic settings.

GradualScale.fred

This script scales your images. You can specify the starting and ending scale settings.

GrayToColor.fred

This script converts grayscale images to raw color data. If it is already color data, it will be skipped.

Highlight.fred

This script highlights a rectangular area of each image. The parts outside of this area will be dimmed.

HistEqualization.fred

This script applies a histogram equalization on your images.

HorizontalFlip.fred

This script flips each image horizontally about the Y (vertical) axis.

HorizontalScroll.fred

This script gradually scrolls an image horizontally off the screen. You can also choose to wrap the image.

Interlace.fred

This script assembles two fields into a frame. This is useful if you have non-interlaced images to put on video tape.

LineArt.fred

This script converts the image to a “line art” representation. It is best that the image have a high contrast to look good so you may have to run the **AdjustBalancing.fred** script prior to this script.

MedianFilter.fred

This script applies a median filter operation on your images.

Mosaic.fred

This script creates a square mosaic effect on your images. The square “tiles” don’t change size at all.

Negative.fred

This script makes a negative of your images.

OilPaint.fred

This script applies an oil paint effect on your images.

Rectangle.fred

This script applies the same type of rectangle (user-specified) on your images.

RemIsolatedPxls.fred

This script removes unwanted pixels from an image. Rendered image data must exist for this operation to succeed, so include the **RenderAsDefined.fred** script before this script.

RenderedToRaw.fred

This script copies the rendered image data to the raw data, losing the previous raw image data.

Roll.fred

This script scrolls the image off in a specified direction, with the ability to wrap the “scrolled”-off parts to the “exposed” parts. The same amount of scrolling will be performed for each image.

Saturation.fred

This script adjusts the color saturation on your images.

ScaleToDoubleSize.fred

This script doubles the width and height of your images. As this creates larger size images, you must have enough memory allocated for the program to store the larger images.

ScaleToHalf.fred

This script scale your images to half the original width and height.

ScaleToPixelAspect.fred

This script scales images to a specific pixel aspect—very useful when transferring a bunch of files between platforms with different pixel aspects.

ScaleToSize.fred

This script scales your images to a specific pixel size.

SimpleRotate.fred

This script incrementally rotates a sequence of images about the center of the image. The entire image is rotated.

SimpleTwirl.fred

This script incrementally twirls a sequence of images. Note that the twirl effect will be centered on the image to encompass the entire image.

SimPrint.fred

This script applies the simulated print operation on your images. Note that this creates rendered data and does not change the raw data.

Tile.fred

This script makes a tile of a rectangular area on your images.

VerticalFlip.fred

This script flips each image vertically about the X (horizontal) axis.

VerticalScroll.fred

This script gradually scrolls an image vertically off the screen. You can also choose to wrap the image.

12.5.2 FREDRenderers

DisplayRendered.fred

This script displays the contents of the render screen for a specified amount of time.

ReDisplay.fred

This script simply redisplay the render screen or window for a specified number of seconds. The script is useful for previewing changes made by adjusting the color balancing of an image. Rendered data must already exist.

RenderAsDCTV.fred

This script allows you to render your images to be displayed properly on a DCTV-equipped machine. DCTV images have a minimum width requirement, so be sure to scale your images if needed before this script.

RenderAsDefined.fred

This script allows you to render your images in whatever screen mode, number of colors, and dither type you select.

If you have other scripts which require rendered data, place this script before them. A very common example is the **SaveAsANIM.fred** script. If you will be creating animations from your images, you will be using these two scripts a lot.

Although you can render to HAM or HAM8 modes with this script, you probably should use the more specific **RenderAsHAM.fred** script to avoid unnecessary questions.

RenderAsHAM.fred

This script allows you to render your images in either HAM or HAM8 mode.

ReRender.fred

This script rerenders your raw image data, creating rendered image data.

12.5.3 FREDsavers

FieldBuster.fred

This script splits a frame into separate odd and even fields.

FrameBuilder.fred

This script builds a full frame from two field images (entered into a FRED sequence as adjacent cells).

OverrideLength.fred

This script is used to force an **InvokeADPro** session that uses the **SaveAsANIM.fred** script (or any other script which pays attention to the **Length** parameter) and is being run with the **Invoke Scripts:** cycle gadget set to **Once Per Frame** to work properly (i.e., each modified frame will only be saved out once). This situation can arise when you have cells that are not of

length 1, but you want your **SaveAsANIM** (or similar functioning) script to work properly.

If you use this script, then be sure to set the **Invoke Scripts:** cycle gadget to **Once Per Frame** and insert this script just before the script (such as **SaveAsANIM**) that needs to be overridden.

At this time, this script is hard-coded to set the **FREDOVERRIDELENGTH** to 1.

SaveAsALIAS.fred

This script saves images in the ALIAS format used by Alias Research products.

SaveAsANIM.fred

This script saves images to an Amiga IFF-ANIM file. ANIM files do not support 24-bit data, so be sure you insert a script which creates rendered data before this script.

SaveAsBMP.fred

This script saves images in Windows BMP format.

SaveAsDPIIE.fred

This script saves images in DPIIE format.

SaveAsFRAMESTORE.fred

This script saves images in Toaster 2.0 (or later) Framestore format. Files in Framestore format should be exactly 752x482 or they may not be displayed properly on the Toaster. If the image is smaller than 752x482 it could be centered and composited on a backdrop that has a resolution of 752x482.

SaveAsGIF.fred

This script saves images in GIF format. Keep in mind that the GIF format supports a MAXIMUM of 256 colors.

SaveAsIFF.fred

This script saves images in Amiga IFF-ILBM format.

SaveAsIMPULSE.fred

This script saves images in IMPULSE RGB8 or RGBN format.

SaveAsJPEG.fred

This script saves images in JPEG format.

SaveAsPCX.fred

This script saves images in PCX format.

SaveAsPICT.fred

This script saves images in PICT format.

SaveAsQRT.fred

This script saves images in QRT format.

SaveAsRENDITION.fred

This script saves images in RENDITION format.

SaveAsSCULPT.fred

This script saves images in SCULPT format.

SaveAsSGI.fred

This script saves images in SGI RGB format.

SaveAsTARGA.fred

This script saves images in TARGA format.

SaveAsTIFF.fred

This script saves images in TIFF format.

SAveAsWAVEFRONT.fred

This script saves images in Wavefront RLA format.

SaveAsX.fred

This scripts saves images in X format, either Window Dump or Bitmap Dump.

SaveToA2410.fred

This script saves images to Commodore's A2410 graphics board.

SaveToDP4ANIM.fred

This script saves images to the DPaint IV AGA animation buffer, thereby directly creating an animation within DPaint.

SaveToFC24.fred

This script saves images to the Impulse FireCracker 24 graphics board.

SaveToFRAMEBUFFER.fred

This script saves images to the Mimetics FrameBuffer.

SaveToOPALVISION.fred

This script saves images to the OpalVision graphics board from Opal Technology / Centaur Development.

SaveToPREFPRINTER.fred

This script saves images to the printer supported by a Preferences printer driver that supporting strip printing.

12.5.4 FREDFunctions

CheckForGrayImageData

This script checks for the existence of gray raw or rendered image data. If it doesn't exist, it tries to convert whatever color data is available to grayscale.

CheckForImageData

This script checks for any type of image data (raw or rendered) in the primary image buffer. Most of the time you will want to be more specific about the type of data you're requiring.

CheckForRawImageData

This script simply checks for the existence of raw image data in the primary image buffer.

CheckForRenderedImageData

This script simply checks for the existence of rendered image data in the primary image buffer.

FileOnly

This script takes a filename and returns just the file portion of it. The file can be concatenated to the result from the same filename being passed to the PathOnly FREDFunction.

GetABool

This script allows you to ask the user to select between two choices. The text for the title bar and **Ok** and **Cancel** buttons must be specified.

GetADir

This script allows you to ask the user to choose a directory name. The text for the title bar, default directory selection, and whether a selection is required must be specified.

GetAFile

This script allows you to ask the user to choose a filename. The text for the title bar, default directory and filename selection, and whether a selection is required must be specified.

GetAFloat

This script allows you to ask the user to choose a floating point number. The text for the title bar, default value, minimum value, maximum value, and whether a value is required must be specified.

GetAListOfFiles

This script allows you to ask the user to choose one or more filenames. The text for the title bar, default directory and filename, and whether a selection is required must be specified.

GetANumber

This script allows you to ask the user to choose a whole number. The text for the title bar, default value, minimum value, maximum value, and whether a number is required must be specified.

GetAString

This script allows you to ask the user to choose a text string. The text for the title bar, default text string, and whether a string is required must be specified.

GetLoader

This script allows you to check for and set the current load format. It is mainly useful for confirming that a specific loader exists.

GetModeID

This script displays the screen mode requester for the selection of a screen mode. The information about the selected mode is returned.

GetNumExtension

This script separates the number from a FRED-created filename.

GetSaver

This script allows you to check for and set the current save format. It is mainly useful for confirming that a specific saver exists.

ModifyFilename

This script performs the filename modification specified in the ModifyFilenameSelect script. As such, it should be called after the call to ModifyFilenameSelect.

ModifyFilenameSelect

This script allows you to choose the type of filename modification that will be performed. It is used in conjunction with and before the ModifyFilename script.

PathOnly

This script takes a filename and returns just the path portion of it. Any trailing colon or forward slash will be returned as well, so that it can be concatenated with the result from the FileOnly FREDFunction.

SeqFileAppend

This script appends the given filename and image information to the given FRED sequence file. As such, this script should be called after calling SeqFileSelect.

SeqFileSelect

This script allows you to specify a FRED sequence file to create for each time the SeqFileAppend script is called. As such, this script should be called before any call to SeqFileAppend.

WordWrap

This script will reformat a text string so that it appears within the given row and column restrictions.

Chapter 13

Reference

This chapter describes all the control panels, menu items, keyboard equivalents and shortcuts, and ARexx commands available in ADPro.

13.1 ADPro Preferences

The ADPro program can be invoked by either double-clicking on its Workbench icon or typing its name at a Shell prompt. You can also customize various parts of the program by specifying some Tool Types or using a few command line arguments.

13.1.1 Public Screen Name

ADPro's public screen name is "ADPro". This cannot be changed.

13.1.2 Tool Type Options

You can specify Tool Type options (using ADPro's icon) to control ADPro. These include:

BEHIND

If present, the screen that ADPro will open on will be opened behind all other screens open at the time.

PUBSCREEN=*public_screen_name*

If present, ADPro will try to open on the public screen with the given name. If that screen doesn't exist or if it fails to open, then it will open on the default public screen. Note that the first 10 characters in the given public screen name will be used to find a screen that matches.

If not present, ADPro opens on the default public screen.

MINIMIZED

If present, ADPro will open its user interface in its minimized state.

If not present, then it will default to either the last saved setting or in its "maximized" state, if no previous settings exist.

LISTINTERFACE

If present, ADPro will open in List Interface mode.

If not present, then it will default to either the last saved setting or the Button Interface, if no previous settings exist.

MAXMEM=*n*

If present, ADPro will be limited to using *n* bytes for its image buffer size. Do not include comma separators in the *n* value. If there is less than *n* bytes free, the buffer size will be equal to the available number, not *n*.

FASTMEMONLY

If present, ADPro will be restricted to only using fast (non-chip/graphics) memory for its image buffer. This is very useful for Amiga users whose machines have equal amounts of chip and fast memory. In this type of setup previous versions of ADPro would likely use chip memory for its image buffer, leaving little or no memory available to display the images being loaded.

If this option is not defined, then the program will behave as before—using the largest contiguous chunk of memory (whether chip or fast) for its image buffer.

APPSCRIPT=filename

If present, ADPro will execute this ARexx script when one or more files are dropped onto ADPro's window (an AppWindow), when it's running on the Workbench screen.

If not present, then no script will be executed.

The list of filenames dropped onto the window are passed to this script (each one quoted and separated by a space character). They can be retrieved with a **PARSE ARG** command at the beginning of the script, such as: **PARSE ARG Filenames**

ENHANCED

If present, the enhanced palette option will be enabled.

NOTE This is inversely equivalent to the old **NOENHANCED** tool type (which is still supported), although we suggest you use this new form.

SETTINGS=filename

If present, the defaults will be loaded from the *filename* specified.

NOTE This is equivalent to the old **DEFAULTFILE** tool type (which is still supported), although we suggest you use this new form.

NOLOADSETTINGS

If present, the **ADPro.prefs** file will not be loaded when ADPro is started.

NOTE This is equivalent to the old **NOLOADDEFAULTS** tool type (which is still supported), although we suggest you use this new form.

NOSAVESETTINGS

If present, the **ADPro.prefs** file will not be saved when ADPro exists.

NOTE This is equivalent to the old **NOSAVEDEFAULTS** tool type (which is still supported), although we suggest you use this new form.

STARTUP=*filename*

If defined, ADPro will execute the ARexx script (named *filename*) immediately after booting up. The *filename* argument must include the full path to the script.

NOTE This is equivalent to the old **STARTSCRIPT** tool type (which is still supported), although we suggest you use this new form.

REXXDIR=*directory*

If defined, ADPro will use the defined *directory* to search for the scripts that are bound to the function, numeric keypad, and main keyboard keys. If not defined, it will search for them in your REXX: directory.

CX_POPKEY=*hot_key_sequence*

If defined, the *hot_key_sequence* will be used to activate ADPro's main window. This allows you to access ADPro's controls with the press of a few keys.

If not defined, the default sequence of **Alt + Shift + a** will be used.

The valid key sequences are defined in your AmigaDOS User Manual. The currently defined key sequence is displayed in ADPro's title bar.

For your convenience, the ADPro icon contains all of the possible Tool Types listed above. All of the Tool Types can be disabled (by surrounding them with parentheses). To enable a Tool Type, click on it (in the list), remove the parentheses from the Tool Type, press the **Return** key to accept the changes to the Tool Type list, and press the **Save** button to save the changes to the icon.

NOTE The following tool types are not supported anymore and will be ignored if used: **SHALLOW**, **ADPROSHALLOW**, **VISUALSHALLOW**, **SUPER72**, **ADPROSUPER72**, and **VISUALSUPER72**.

13.1.3 Command Line (Shell) Arguments

Alternatively, you can use one of more of the Command Line (Shell) options :

BEHIND or **BH**

If present, the screen that ADPro will open on will be opened behind all other screens.

PUBSCREEN=*public_screen_name* or **PS=***public_screen_name*

If present, ADPro will try to open on the public screen with the given name. If that screen doesn't exist or if it fails to open, then it will open on the default public screen. Note that the first 10 characters in the given public screen name will be used to find a screen that matches.

If not present, ADPro opens on the default public screen.

MINIMIZED

If present, ADPro will open its user interface in its minimized state.

If not present, then it will default to either the last saved setting or in its “maximized” state, if no previous settings exist.

LISTINTERFACE or LI

If present, ADPro will open in List Interface mode.

If not present, then it will default to either the last saved setting or the Button Interface, if no previous settings exist.

MAXMEM=*n* or MM=*n*

If present, ADPro will be limited to using *n* bytes for its image buffer size. Do not include comma separators in the *n* value. If there is less than *n* bytes free, the buffer size will be equal to the available number, not *n*.

FASTMEMONLY

If present, ADPro will be restricted to only using fast (non-chip/graphics) memory for its image buffer. This is very useful for Amiga users whose machines have equal amounts of chip and fast memory. In this type of setup previous versions of ADPro would likely use chip memory for its image buffer, leaving little or no memory available to display the images being loaded.

If this option is not defined, then the program will behave as before—using the largest contiguous chunk of memory (whether chip or fast) for its image buffer.

APPSCRIPT=*filename* or AS=*filename*

If present, ADPro will execute this ARexx script when one or more files are dropped onto ADPro’s window (an AppWindow), when it’s running on the Workbench screen.

If not present, then no script will be executed.

The list of filenames dropped onto the window are passed to this script (each one quoted and separated by a space character). They can be retrieved with a **PARSE ARG** command at the beginning of the script, such as: **PARSE ARG Filenames**

ENHANCED or EN

If present, the enhanced palette option will be enabled.

NOTE This is inversely equivalent to the old **NOENHANCED** option (which is still supported), although we suggest you use this new form.

DEFAULTFILE=*filename* or DF=*filename*

If present, the defaults will be loaded from the *filename* specified.

NOLOADSETTINGS or NL

If present, the **ADPro.prefs** file will not be loaded when ADPro is started.

NOTE This is equivalent to the old **NOLOADDEFAULTS** option (which is still supported), although we suggest you use this new form.

NOSAVESETTINGS or NS

If present, the **ADPro.prefs** file will not be saved when ADPro exits.

NOTE This is equivalent to the old **NOSAVEDEFAULTS** option (which is still supported), although we suggest you use this new form.

STARTSCRIPT=*filename* or SS=*filename*

If defined, ADPro will execute the ARexx script (named *filename*) immediately after booting up. The *filename* argument must include the full path to the script.

NOTE This is equivalent to the old **STARTSCRIPT** option (which is still supported), although we suggest you use this new form.

REXXDIR=*directory* or RD=*directory*

If defined, ADPro will use the defined *directory* to search for the scripts that are bound to the function, numeric keypad, and main keyboard keys. If not defined, it will search for them in your REXX: directory.

NOTE The following options are not supported anymore and will be ignored if used: **SHALLOW**, **ADPROSHALLOW**, **VISUALSHALLOW**, **SUPER72**, **ADPROSUPER72**, and **VISUALSUPER72**.

13.2 ADPro Keyboard Shortcuts

List Interface	
Key Sequence	Function
L	Place focus on L oaders list
S	Place focus on S avers list
O	Place focus on O perators list
U	Place focus on U ser C ommands list
Tab	Activate next list
Shift + Tab	Activate previous list

Button Interface	
Key Sequence	Function
L	Open/activate L oaders list
S	Open/activate S avers list
O	Open/activate O perators list
U	Open/activate U ser C ommands list

Floating Module Lists	
Key Sequence	Function
↑/↓	scroll the list by one entry up/down
Alt + ↑/↓	scroll half of previous/next view
Shift + ↑/↓	scroll previous/next view
Ctrl + ↑/↓	scroll to top/bottom of list
Esc	close list window

Render Screen/Window	
Key Sequence	Function
←/→	scroll by larger of width/5 or 64 pixels
↑/↓	scroll by larger of height/5 or 40 pixels
Alt + ←/→	scroll by half of ←/→ distance
Alt + ↑/↓	scroll by half of ↑/↓ distance
Shift + ←/→	scroll by width of displayed image left/right
Shift + ↑/↓	scroll by height of displayed image up/down
Ctrl + ←/→	scroll to leftmost/rightmost part of image
Ctrl + ↑/↓	scroll to topmost/bottommost part of image
Keypad 7	scroll up and to the left
Keypad 8	scroll up
Keypad 9	scroll up and to the right
Keypad 6	scroll right
Keypad 3	scroll down and to the right
Keypad 2	scroll down
Keypad 1	scroll down and to the left
Keypad 4	scroll left
Space bar	toggle between render screen and main window
Esc	close the render screen/window

Compositing Screen	
Key Sequence	Function
←/→	scroll one pixel left/right
↑/↓	scroll one pixel up/down
Alt + ←/→	scroll half of source width left/right
Alt + ↑/↓	scroll half of source height up/down
Shift + ←/→	scroll source width left/right
Shift + ↑/↓	scroll source height up/down
Ctrl + ←/→	align left/right edges
Ctrl + ↑/↓	align top/bottom edges
I	toggle size of Compositing panel

13.3 ADPro Menus

This section gives a brief description of each menu item available in the ADPro menu bar.

13.3.1 Project Menu

Clear erases the raw and rendered image data from memory. Note that no warning that you will lose your image in memory is given.

Open... launches the UNIVERSAL loader to import image data into ADPro.

Save As... launches the IFF saver to export the current image data to a file on disk.

Print As... launches the PREFPRINTER saver to print the current raw image data to the currently selected (from the Printer Preferences editor) printer. Note that the selected printer driver **MUST** support strip printing for PREFPRINTER to work with it.

Release Notes... displays the text file describing last minute changes to the program, its utilities, and documentation. The file it displays is "ADPRO:ADPro.notes".

About... displays information about the program, including version number, serial number, primary image buffer size (in bytes), temporary buffer status indicator, ARexx port name, copyright information, and text asking for your help in keeping Art Department Professional a professionally supported product.

Quit ADPro exits the program.

13.3.2 Edit Menu

Copy launches the CLIPBOARD saver to export IFF-ILBM image data to the Amiga's Clipboard.

Paste launches the CLIPBOARD loader to import IFF-ILBM image data from the Amiga's Clipboard.

Clear Clipboard erases the contents of the Amiga's Clipboard, regardless of the contents.

13.3.3 Loaders Menu

The first three menu items show your previously-selected favorite loaders.

Open Loader List... displays the floating Loaders list. This menu item changes to **Close Loader List** when this floating list is open. Selecting **Close Loader List** removes the floating list from view, changing this menu item back to **Open Loader List...**

The bottom four menu items control how the next loaded image will be loaded. **Replace** overwrites (replaces) the current image in the primary image buffer with the loaded image. **Compose** allows you to create a composite between the current and loaded images. **Landscape** overwrites the current image, but also rotates the incoming image 90 degrees counter-clockwise. **As Alpha** loads the incoming image into "alpha memory" for use in a future composite operation.

13.3.4 Savers Menu

The first three menu items show your previously-selected favorite savers.

Open Saver List... displays the floating Savers list. This menu item changes to **Close Saver List** when this floating list is open. Selecting **Close Saver List** removes the floating list from view, changing this menu item back to **Open Saver List...**

13.3.5 Operators Menu

The first three menu items show your previously-selected favorite operators.

Open Operator List... displays the floating Operators list. This menu item changes to **Close Operator List** when this floating list is open. Selecting

Close Operator List removes the floating list from view, changing this menu item back to **Open Operator List...**

13.3.6 Display Menu

Balancing... displays the Balancing control panel, from which color balancing adjustments can be performed.

Palette... displays the Palette control panel, from which palette adjustments can be performed.

The **Dither** submenu controls the type of dither to use. Available choices include: **Off**, **Floyd**, **Burkes**, **Sierra**, **Jarvis**, **Stucki**, **Random**, **Large Ordered**, and **Small Ordered**.

Dither Amount (*n*)... displays the Dither Amount control panel, from which a dither amount can be specified (this number will show up in the menu item in place of the *n*). This menu item can only be selected if the **Random**, **Large Ordered**, or **Small Ordered** dither is selected.

Redisplay displays the image on the currently defined render screen.

Execute rerenders the raw image data through the current rendering settings and displays it on the currently defined render screen.

Set Render Screen... displays the Set Render Screen requester, from which a render screen (or window) may be selected.

Close Render Screen removes the open render screen (or window) from view.

13.3.7 Settings Menu

Set ADPro's Screen... displays a Screen Mode requester, from which a screen may be selected. ADPro's window will open on this screen.

Set ADPro's Font... displays the ASL font requester, from which a font can be selected for ADPro's screen. Note that this command will only be available when ADPro is running on its own screen.

The **Set Visual** submenu contains commands for specifying the screen mode to use for the visual operators. **Screen Mode...** displays a screen mode requester, from which the mode can be selected. **Deep** toggles whether it will open in as a 3 bit-plane/8 shades (not selected) or 8 bit-plane/256 shade (selected) screen.

Minimized toggles ADPro's window between its visible form to its minimized

form, which is as a small window title bar.

Button Interface toggles ADPro's window between the Button and List Interfaces.

The **Set Modules** submenu lets you assign the currently selected module (loader, saver, or operator) in one of the floating module lists to a corresponding "favorite" button (**Module #1**, **Module #2**, or **Module #3**) in the Button Interface.

Locked Screen Mode toggles whether the next image to be loaded will change (unchecked) or use (checked) the current screen mode defined in the Screen Information area.

Locked Depth toggles whether the next image to be loaded will change (unchecked) or use (checked) the current render depth defined in the Screen Information area.

Available Modes Only toggles whether you will (checked) or will not (unchecked) be limited to the screen modes and depths supported by your Amiga hardware. If you will be creating images for use on other people's machines (with display hardware can support more modes, this menu item should not be checked).

Load Settings... displays a file requester, from which you can select a previously-saved ADPro settings file to load and use.

Save Settings stores the current program settings using the settings filename used when starting ADPro. If no filename was specified, then it will use "ADPro.prefs".

Save Settings As... displays a file requester, from which the current program settings can be saved under a new filename.

13.3.8 User Menu

The first three menu items show your previously-selected favorite user commands.

Open User Command List... displays the floating User Commands list. This menu item changes to **Close User Command List** when this floating list is open. Selecting **Close User Command List** removes the floating list from view, changing this menu item back to **Open User Command List...**

Execute ARexx Script... displays a control panel, from which an ARexx script can be executed.

13.4 General Controls and Information

The following ARexx commands pertain to the information available on ADPro's main control window and to ADPro's user interface.

13.4.1 Version Number

Syntax: **VERSION**

This command returns the version identification string of the currently executing ADPro.

Operation	RC	ADPRO_RESULT
always returns	0	version string

13.4.2 Serial Number

Syntax: **SERIAL_NUMBER**

This command returns the serial number of this copy of ADPro in **ADPRO_RESULT**. The serial number can be found in the About panel, which can be displayed by selecting the **About...** menu item under the **Project** menu. This command will never return an error.

Operation	RC	ADPRO_RESULT
always returns	0	serial number

13.4.3 Image Buffer Size

Syntax: **CURRENT_MEM_SIZE**

This command will return the number of bytes of main memory in use by ADPro's primary image memory buffer. This value is returned in **ADPRO_RESULT**.

This value is reported from the user interface in the **About** panel.

Operation	RC	ADPRO_RESULT
always returns	0	image buffer size

13.4.4 Exiting ADPro

Syntax: **ADPRO_EXIT**

This command causes ADPro to terminate. Clearly, any ARexx command destined for ADPro executed after this command will not work, since ADPro will no longer be present to service it.

Operation	RC	ADPRO_RESULT
always returns	0	unchanged

13.4.5 Specifying ADPro's Screen

Syntax: **SET_ADPRO_MODE** (1)
SET_ADPRO_MODE *screen_id width height colors* (2)

The first form returns the current mode information.

The second form sets the type of screen on which to open the ADPro window. The parameters that are required are the same type as those returned by the **GET_SCREEN_MODE** command.

NOTE The screen that is specified must be at least 640 x 200 to be valid.

If	RC	ADPRO_RESULT
successful	0	undefined
<i>width, height, or colors</i> are detectable illegal	10	"Invalid argument to function"
screen couldn't be opened	10	"Couldn't open screen"

13.4.6 Specifying on Which Public Screen to Open ADPro

Syntax: **SET_ADPRO_PUBLIC** (1)
SET_ADPRO_PUBLIC *public_screen_name* (2)

The first form returns the name of the public screen on which the ADPro window is open.

The second form sets on which public screen the ADPro window will open. The parameters that are required are the same type as those returned by the **GET_SCREEN_MODE** command.

This command differs from the **SET_ADPRO_MODE** command in that **SET_ADPRO_MODE** allows you to select a non-public screen to open on, whereas **SET_ADPRO_PUBLIC** allows you to select a public screen.

If	RC	ADPRO_RESULT
successful	0	previous public screen name
public screen could not be found	10	"Couldn't find public screen"

This command will return "ADPro" if ADPro is running on its own screen.

13.4.7 Bringing the ADPro Screen to Front

Syntax: **ADPRO_TO_FRONT**

This command causes the screen containing ADPro's window to pop in front of all other screens.

Operation	RC	ADPRO_RESULT
always returns	0	undefined

13.4.8 Pushing the ADPro Screen Behind

Syntax: **ADPRO_TO_BACK**

This command causes the screen containing ADPro's window to be pushed behind all other screens.

Operation	RC	ADPRO_RESULT
always returns	0	undefined

13.4.9 Bringing a Public Screen to Front

Syntax: **SCREEN_TO_FRONT** *public_screen_name*

This command causes the named public screen to pop in front of all other screens. If the public screen name is more than 10 characters long, only the first 10 will be examined for a match.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	"Error"

13.4.10 Specifying the Type of ADPro's Interface

Syntax: **INTERFACE** (1)
INTERFACE (BUTTON | LIST) (2)

This command is equivalent to the **Button Interface** command under the **Settings** menu.

The first form returns what interface (**BUTTON** or **LIST**) ADPro is currently using.

The second form sets ADPro's interface to the given type.

If operation was	RC	ADPRO_RESULT
successful	0	the <i>previous</i> interface type
not successful	10	"Invalid Argument To Function"

13.4.11 Specifying the Size of ADPro's Interface

Syntax: **INTERFACE_SIZE** (1)

INTERFACE_SIZE (**TINY** | **LARGE**) (2)

This command is the equivalent of the **Minimized** command under the **Settings** menu.

The first form returns the size (**TINY** or **LARGE**) of ADPro's interface.

The second form sets ADPro's interface to the given size.

If operation was	RC	ADPRO_RESULT
successful	0	the <i>previous</i> interface size
not successful	10	"Invalid Argument To Function"

13.5 Balancing Controls

These commands interrogate or set values affecting color balancing.

13.5.1 Red Adjustment

Syntax: **RED** (1)

RED *value* (2)

The first form returns the current red adjustment setting.

The second format sets the red adjustment to the supplied value, which must be in the range of -50 to 50.

If value was	RC	ADPRO_RESULT
accepted	0	the <i>previous</i> red adjustment setting
out of range	10	"Invalid Argument To Function"

Note that if the value you wish to specify is a negative number, you must surround the negative number with quotation marks.

See Section 5.7.2 for more information about the red adjustment.

13.5.2 Green Adjustment

Syntax: **GREEN** (1)
GREEN *value* (2)

The first form returns the current green adjustment setting.

The second format sets the green adjustment to the supplied value, which must be in the range of -50 to 50.

If value was	RC	ADPRO_RESULT
accepted	0	the <i>previous</i> green adjustment setting
out of range	10	"Invalid Argument To Function"

Note that if the value you wish to specify is a negative number, you must surround the negative number with quotation marks.

See Section 5.7.2 for more information about the green adjustment.

13.5.3 Blue Adjustment

Syntax: **BLUE** (1)
BLUE *value* (2)

The first form returns the current blue adjustment setting.

The second format sets the blue adjustment to the supplied value, which must be in the range of -50 to 50.

If value was	RC	ADPRO_RESULT
accepted	0	the <i>previous</i> blue adjustment setting
out of range	10	"Invalid Argument To Function"

Note that if the value you wish to specify is a negative number, you must surround the negative number with quotation marks.

See Section 5.7.2 for more information about the blue adjustment.

13.5.4 Brightness Adjustment

Syntax: **BRIGHTNESS** (1)
BRIGHTNESS *value* (2)

The first form returns the current brightness adjustment setting.

The second format sets the brightness adjustment to the supplied value, which must be in the range of - 50 to 50.

If value was	RC	ADPRO_RESULT
accepted	0	the <i>previous</i> brightness adjustment setting
out of range	10	"Invalid Argument To Function"

Note that if the value you wish to specify is a negative number, you must surround the negative number with quotation marks.

See Section 5.7.3 for more information about the brightness adjustment.

13.5.5 Contrast Adjustment

Syntax: **CONTRAST** (1)
CONTRAST *value* (2)

The first form returns the current contrast adjustment setting.

The second format sets the contrast adjustment to the supplied value, which must be in the range of -50 to 50.

If value was	RC	ADPRO_RESULT
accepted	0	the <i>previous</i> contrast adjustment setting
out of range	10	"Invalid Argument To Function"

Note that if the value you wish to specify is a negative number, you must surround the negative number with quotation marks.

See Section 5.7.4 for more information about the contrast adjustment.

13.5.6 Gamma Adjustment

Syntax: **GAMMA** (1)
GAMMA *value* (2)

The first form returns the current gamma adjustment setting.

The second format sets the gamma adjustment to the supplied value, which must be in the range of -50 to 50.

If value was	RC	ADPRO_RESULT
accepted	0	the <i>previous</i> gamma adjustment setting
out of range	10	"Invalid Argument To Function"

NOTE Although you can set a gamma value in the range of -50 to 50 in the user interface (the **Gamma:** slider and input field), the ARexx interface must be specified in the range of 0 to 100, where 50 means no gamma correction.

See Section 5.7.5 for more information about the gamma adjustment.

13.6 Palette Controls

The commands in this subsection refer to and manipulate the settings in the Palette control panel.

13.6.1 Palette Width

Syntax: PWIDTH (1)

PWIDTH (NORMAL | ENHANCED) (2)

The first form of the command returns the current width (accuracy) of the palette in the ADPRO_RESULT variable.

The second form of the command allows the ARexx programmer to set the palette width to the supplied value, which must be either NORMAL or ENHANCED.

If value was	RC	ADPRO_RESULT
either NORMAL or ENHANCED	0	the <i>previous</i> palette depth
neither NORMAL nor ENHANCED	10	"Invalid Argument To Function"

Note then when the palette is locked and you switch from ENHANCED to NORMAL, some resolution is lost. Switching back does not regain this lost resolution. When the palette is unlocked, palette recomputation will be done at the current palette width.

For more information concerning palette width, please refer to Section 5.8.5.

13.6.2 Palette Status

Syntax: **PSTATUS** (1)
PSTATUS (LOCKED | UNLOCKED) (2)

The first form returns the string **LOCKED** or **UNLOCKED** in the **ADPRO_RESULT** variable, provided that results have been requested.

The second form sets the status of the palette to the specified value, which must be the string **LOCKED** or **UNLOCKED**.

If value was	RC	ADPRO_RESULT
either LOCKED or UNLOCKED	0	the <i>previous</i> state of the palette
neither LOCKED nor UNLOCKED	10	"Invalid Argument To Function"

For more information about palette status, refer to Section 5.8.4.

13.6.3 Total Number of Colors

Syntax: **PTOTAL** (1)
PTOTAL value (2)

The first form returns the currently defined total number of colors in **ADPRO_RESULT** provided that results have been requested.

The second form sets the total number of colors to the supplied value which must be one of the following: 2, 4, 8, 16, 32, 64, 128, 256, **EHB**, **HAM**, or **HAM8**.

When specifying **HAM**, the actual total number of colors possible refers to the number of color registers available in the standard **HAM** mode (which is 16). Similarly, when referring to an **EHB** screen, the actual total number of colors available is 32.

If value was	RC	ADPRO_RESULT
accepted	0	total number of colors
out of range	10	"Invalid Argument To Function"

Note that the total number of colors, the offset of color zero, and the number of colors to be used must be consistent. For example, an error will be generated if you request a total number of colors which is smaller than the currently defined number of colors used.

For more information about the total number of colors, please see Section 5.8.2.

13.6.4 Number of Colors Used

Syntax: `PUSED` (1)
`PUSED value` (2)

The first form returns the current number of colors to be used in `ADPRO_RESULT` provided that results have been requested.

The second form sets the number of colors to be used to the supplied value which must range between 2 and the total number of colors.

If value was	RC	ADPRO_RESULT
accepted	0	the <i>previous</i> number of colors used
out of range	10	"Invalid Argument To Function"

Note that the total number of colors, the offset of color zero, and the number of colors to be used must be consistent. For example, an error will be generated if you request a number of colors to be used which is larger than the total number of colors.

For more information about the number of colors to be used, please see Section 5.8.2.

13.6.5 Offset Color Zero

Syntax: `POFFSET` (1)
`POFFSET value` (2)

The first form returns the currently defined offset of color zero in `ADPRO_RESULT` provided that results have been requested.

The second form sets the offset of color zero to the supplied value, which must range from 0 to the total number of colors minus one. When in `HAM` mode, the maximum value for the offset of color zero is 14. When in `HAM8` mode, the maximum value for the offset of color zero is 62. When in `EHB` mode, the maximum is 30.

If value was	RC	ADPRO_RESULT
accepted	0	offset value
out of range	10	"Invalid Argument To Function"

Note that the total number of colors, the offset of color zero, and the number of colors to be used must be consistent. For example, an error will be generated if you request an offset of color zero which would overflow the total number of colors presently defined when the number of colors to be used is factored in.

For more information about the offset of color zero, please refer to Section 5.8.2.

13.6.6 Custom Palette Values

Syntax: **SETPALINFO** (1)
SETPALINFO *total used offset* (2)

This command lets you specify the **PTOTAL**, **PUSED**, and **POFFSET** values, *but in one command invocation instead of three.*

The first form returns in **ADPRO_RESULT** a string containing:

total used offset

where:

total = **Colors (Total):** value;
used = **Colors (Used):** value;
offset = **Offset Color Zero:** value

The second form sets the *total*, *used*, and *offset* values.

If operation was	RC	ADPRO_RESULT
successful	0	<i>total used offset</i>
not successful	10	"Error"

13.6.7 Palette Contrast

Syntax: **PCONTRAST** (1)
PCONTRAST (0 | 1) (2)

The first form of the command returns the present setting of the **Color Ordering:** cycle gadget. If **ADPRO_RESULT** contains a 0, then the gadget is set to **No Contrast**. Otherwise, it is set to **Contrast**.

The second form allows you to set the state of the cycle gadget. If you supply an argument of 0, then the gadget is set to the off or **No Contrast** setting. If you supply an argument of 1, then the gadget is set to the on or **Contrast** setting.

If value was	RC	ADPRO_RESULT
either 0 or 1	0	undefined
neither 0 nor 1	10	undefined

For more information about the **Color Ordering:** cycle gadget, please refer to Section 5.8.3.

13.6.8 Palette Sort Order

Syntax: PSORT (1)
PSORT (0 | 1) (2)

The first form of the command returns the present palette sort order. If **ADPRO_RESULT** contains a 0, then the sort order is ascending. Otherwise, the sort order is descending.

The second form sets the palette sort order to ascending or descending depending upon whether the supplied argument is a 0 or 1, respectively.

If value was	RC	ADPRO_RESULT
either 0 or 1	0	the <i>previous</i> sort direction
neither 0 nor 1	10	"Invalid Argument To Function"

For more information concerning palette sorting, please refer to Section 5.8.3.

13.6.9 Individual Palette Contents

Syntax: PPOKE *register* (1)
PPOKE *register red green blue* (2)

The first form of the command returns a string describing the contents of the specified color register. The string, returned in **ADPRO_RESULT**, is formatted as the red, green, and blue values of the color register separated by spaces and provided in decimal in the range of 0 to 255.

The second form of the command allows you to set the contents of the specified color register with the supplied values. Range checking will be performed on the supplied color values, which should range from 0 to 255.

For both forms, range checking will be performed on the supplied register number.

If operation was	RC	ADPRO_RESULT
successful	0	"red green blue"
not successful	non-zero	string indicating the nature of the error

While the allowable range of color values is from 0 to 255 (24 bit accuracy), ADPro's internal storage of the colors you define may vary according to whether you are in a normal or enhanced palette mode. As a consequence, it is normal to read a slightly different value than what might have been written immediately before.

You can convert 12 bit colors (the ones you would use with Deluxe Paint, for example) into 24 bit colors by multiplying each color by 17. So, for example, 15 becomes 255 while 1 would become 17.

13.6.10 Multiple Palette Contents

Syntax: **MPPOKE** *register count* (1)

MPPOKE *register count r₁ g₁ b₁...r_{count} g_{count} b_{count}* (2)

The first form returns the *r - g - b* values (*r₁ g₁ b₁...r_{count} g_{count} b_{count}*) in **ADPRO_RESULT**. A space character delimits each value.

The second form returns same as first form, but also sets the values to the specified amounts.

If operation was	RC	ADPRO_RESULT
successful	0	sequence of <i>r - g - b</i> values
not successful	10	<i>previous r - g - b</i> sequence

NOTE If an error occurred, you may be left with partially modified data. ARexx has a limit of 65,535 characters for any string that it sends you. This may affect how much data is received back in **ADPRO_RESULT**.

13.6.11 Setting Pixel Colors

Syntax: **PIXELPOKE** *xoff yoff width height data₀...data_n*

Fill a rectangular area with color or grayscale data. The upper left corner is specified by *xoff* and *yoff*, while the dimensions are specified by *width* and *height*. Each data entry can be *Gray* or *R G B*. Regardless of the current data type, if you specify "-1" for a data item, that item gets ignored.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	"Error"

13.6.12 Getting Pixel Colors

Syntax: **PIXELPEEK** *xoff yoff width height*

Query the contents of a rectangular area. The upper left corner is specified by *xoff* and *yoff*, while the dimensions are specified by *width* and *height*. Each data entry can be *Gray* or *R G B*.

If operation was	RC	ADPRO_RESULT
successful	0	sequence of <i>Gray</i> or <i>R G B</i> values
not successful	10	error message

NOTE ARexx has a limit of 65,535 characters for any string that it sends you. This may affect how much data is received back in **ADPRO_RESULT**.

13.6.13 Getting the Workbench Palette

Syntax: **PGETWB**

This command attempts to load the currently defined Workbench palette.

Note that this command results in the loading of up to four colors only. Should you require access to a deeper Workbench palette (possible under Workbench 2.0 and later), you can use the **PLOAD** command instead.

The Workbench palette is loaded starting with the color register specified by **POFFSET**.

Operation	RC	ADPRO_RESULT
always returns	0	unchanged

For more information about this command, please refer to Section 5.8.3.

13.6.14 Loading a Palette

Syntax: **PLOAD** (1)
PLOAD filename (2)

If no filename is supplied (as in the first form), then ADPro will bring up its file requester so that the user can manually enter the name of a palette file to be loaded. Note that ADPro will automatically pop the screen it's on to front whenever it displays its file requester. Therefore, it is the ARexx programmer's responsibility to depth-arrange screens after calling this function with no supplied filename.

The second form of this command attempts to load a palette from the named file. The supplied file must be in **IFF** format but does not have to contain any image data.

If palette was	RC	ADPRO_RESULT
loaded	0	empty string
not loaded	10	empty string

Note that loading a palette is directly affected by the total number of colors, the number of colors used, and the offset of color zero.

For more information concerning the loading of a palette, please see Section 5.8.3.

13.6.15 Saving a Palette

Syntax: **PSAVE** (1)
PSAVE filename (2)

If a filename is not supplied (as in the first form), ADPro will display its file requester to allow the user to input a filename manually. Remember to handle the case of the user not making any file selection at all.

The second form of this command attempts to save the currently defined palette using the provided filename. The resulting file will be encoded in the IFF format.

If palette was	RC	ADPRO_RESULT
saved	0	empty string
not saved	10	empty string

It is the ARexx programmer's responsibility to prevent the unwanted overwriting of a pre-existing file.

For more information about saving a palette, please refer to Section 5.8.3.

13.6.16 The Wrong Way to Control the Palette

In each of the definitions for **PTOTAL**, **PUSED**, and **POFFSET**, a warning is given that the total number of colors, the number of colors to be used, and the offset of color zero must be consistent.

For example, the following is one of many wrong ways to set the total number of colors to 16 with 5 used and a color zero offset of 3:

```
POFFSET 3
PTOTAL 16
POFFSET 3
PUSED 5
```

The above example will fail if the number of colors to be used had an initial value above 12. This is because setting the offset of color zero to 3 would exceed the total number of colors if the number of colors to be used was 13 or higher.

Pathological cases abound, where the order in which these values are set will determine if an error occurs or not. Setting the offset of color zero to zero (then setting the other desired values) always increases your safety.

Make use of the query versions of the palette commands prior to setting palette values to avoid running afoul of ADPro's internal consistency check ($PUSED + POFFSET \leq PTOTAL$).

13.7 Other ARexx Commands

This section describes commands which are available through ARexx which do not have direct equivalents in the ADPro user interface or don't fall into any of the previously described categories.

13.7.1 Last Saved Image

Syntax: `LAST_SAVED_IMAGE`

This command returns (in `ADPRO_RESULT`) the entire path name used during the last manually executed Save command. A manually executed Save is one *not* performed from ARexx.

If manually performed Save or Load	RC	ADPRO_RESULT
has been performed	0	path name
has not been performed	10	undefined

Note that if the user has not performed a Save by the time the first Load is executed, ADPro will copy the **Load** filename to the **Save** filename. This is the only circumstance in which `LAST_SAVED_IMAGE` will return a filename which wasn't actually saved.

13.7.2 Last Loaded Image

Syntax: `LAST_LOADED_IMAGE`

This command returns (in `ADPRO_RESULT`) the entire path name used during the last manually executed Load command. A manually executed Load is one *not* performed from ARexx.

If manually performed Load or Save	RC	ADPRO_RESULT
has been performed	0	path name
has not been performed	10	undefined

Note that if the user has not performed a Load by the time the first Save is executed, ADPro will copy the Save filename to the Load filename. This is the only circumstance in which `LAST_LOADED_IMAGE` will return a filename which wasn't actually loaded.

13.7.3 Pausing a Specified Time

Syntax: `PAUSE ticks`

This command “kills time” for the specified number of ticks. A tick lasts for one-fiftieth of a second. The specified value must be in the range of 1 to 180000.

If value was	RC	ADPRO_RESULT
accepted	0	unchanged
out of range	10	“Invalid Argument To Function”

13.7.4 Getting a String From the User

Syntax: `GETSTRING` (1)
`GETSTRING title` (2)
`GETSTRING title default` (3)

This command can be used to query the user for a string to be returned into your ARexx program.

The first form of the command displays a string requester with a default title and no default text.

The second form uses the supplied string for the title bar of the string requester.

The third form uses the supplied strings for the title bar of the string requester and the default contents of the string requester, respectively.

If a string was	RC	ADPRO_RESULT
entered	0	the entered string
not entered	10	undefined

It is the ARexx programmer's responsibility to move ADPro to front (using `ADPRO_TO_FRONT`) before calling this function.

Note that strings with spaces must be quoted properly as in the following example:

```
GETSTRING 'Spaces' '"Require Double Quoting!'"
```

13.7.5 Getting an Integer Number From the User

Syntax: `GETNUMBER` (1)
 `GETNUMBER title` (2)
 `GETNUMBER title default` (3)
 `GETNUMBER title default min max` (4)

This command can be used to get a signed integer from the user.

The first form of the command displays an integer requester with a default title and no default value.

The second form uses the supplied title and no default value.

The third form uses the supplied title and the supplied default value.

The fourth form allows you to specify a minimum and maximum value between which the users input must fall.

If a number was	RC	ADPRO_RESULT
entered	0	the entered value
not entered	10	undefined

It is the ARexx programmer's responsibility to move ADPro to front (using `ADPRO_TO_FRONT`) before calling this function.

Note that strings with spaces must be quoted properly as in the following example:

```
GETNUMBER '"Spaces Require Double Quoting!'"
```

13.7.6 Getting a Floating Point Number From the User

Syntax: `GETNUMBER` (1)
 `GETNUMBER title` (2)
 `GETNUMBER title default` (3)
 `GETNUMBER title default min max` (4)

This command can be used to get a signed integer from the user.

The first form of the command displays an integer requester with a default title and no default value.

The second form uses the supplied title and no default value.

The third form uses the supplied title and the supplied default value.

The fourth form allows you to specify a minimum and maximum value between which the users input must fall.

If a number was	RC	ADPRO_RESULT
entered	0	the entered value
not entered	10	undefined

It is the ARexx programmer's responsibility to move ADPro to front (using `ADPRO_TO_FRONT`) before calling this function.

Note that strings with spaces must be quoted properly as in the following example:

```
GETNUMBER '"Spaces Require Double Quoting!'"
```

13.7.7 Getting a Filename From the User

Syntax: `GETFILE` (1)
`GETFILE title` (2)
`GETFILE title default_dir` (3)
`GETFILE title default_dir default_file` (4)

This command can be used to get a filename from the user.

The first form of the command displays the file requester with a default title and no default filename.

The second form uses the supplied title and no default filename.

The third form uses the supplied title and the supplied default directory name. That is, the file requester will load the contents of the specified default directory.

The fourth form allows you to specify a title, a default directory, as well as a default filename.

If a file was	RC	ADPRO_RESULT
selected	0	the selected filename
not selected	10	undefined

It is the ARexx programmer's responsibility to move ADPro to front (using `ADPRO_TO_FRONT`) before calling this function.

Note that strings with spaces must be quoted properly as in the following example:

```
GETFILE '"Spaces Require Double Quoting!'"
```

13.7.8 Getting a List of Filenames From the User

Syntax: **GETFILES** (1)
 GETFILES *title* (2)
 GETFILES *title default_dir* (3)
 GETFILES *title default_dir default_file* (4)

This command can be used to get a list of filenames from the user.

The first form of the command displays the file requester with a default title and no default filename.

The second form uses the supplied title and no default filename.

The third form uses the supplied title and the supplied default directory name. That is, the file requester will load the contents of the specified default directory.

The fourth form allows you to specify a title, a default directory, as well as a default filename.

If	RC	ADPRO_RESULT
at least one file was selected	0	the selected filenames, separated by quotation marks
no file was selected	10	undefined

To select more than one file using the file requester, simply hold down either **Shift** key while clicking on filenames. All filenames selected must be from the same directory. This command makes it easier to write ARexx programs which will batch process multiple files, since it gives you a way of allowing the user to specify multiple files at one time.

It is the ARexx programmer's responsibility to move ADPro to front (using **ADPRO_TO_FRONT**) before calling this function.

Note that strings with spaces must be quoted properly as in the following example:

```
GETFILES '"Spaces Require Double Quoting!'"
```

Following, is an examples of using **GETFILES** and parsing its results.

```
/*  
** An example of using GETFILES.  
*/
```


Note that strings with spaces must be quoted properly as in the following example:

```
GETDIR "Spaces Require Double Quoting!"
```

13.7.10 Getting a Screen Mode From the User

Syntax: `GET_SCREEN_MODE title module_name screen_id width height colors`

This command displays a screen mode requester, from which a screen mode can be selected. The parameters that are given define the default values.

<i>title</i>	This is the text displayed in the requester's title bar.
<i>module_name</i>	This is the name of the display module (in the <code>Displays2</code> directory) to use.
<i>screen_id</i>	This describes the default screen mode; this will be a mode id or public screen name, depending upon which <i>module_name</i> is specified. For example, to default to a window on CygnusEd Professional's first public screen use the following <i>module_name</i> and <i>screen_id</i> pair: "Window CygnusEdScreen1"
<i>width</i>	This is the default value to place in the Width: field. Set this value to 0 to default to the specified <i>screen_id</i> 's width.
<i>height</i>	This is the default for Height: . Set this value to 0 to default to the specified <i>screen_id</i> 's height.
<i>colors</i>	This is the default for Colors: . Possible values are 1, 2, 4, 8, 16, 32, 64, 128, 256, EHB, HAM, HAM8, TRUECOLOR, 32K, 64K, and 16M.

If screen mode was	RC	ADPRO_RESULT
selected	0	the screen mode identification string
not selected	10	undefined

This identification string is of the form:

module_name screen_id width height colors

which is the same string that needs to be sent to the `SET_RENDER_MODE` command.

13.7.11 Retrieve the Entries In Internal Lists

Syntax: `GETLIST (LOADERS | SAVERS | OPERATORS |
MONITORS | COLORS | PALCOLORS | DITHERS)`

This command retrieves a sequence of quoted list entries for various program lists. **LOADERS** will retrieve the Loader modules, **SAVERS** for the Saver modules, **OPERATORS** for the Operator modules, **MONITORS** for the Monitor Types, **COLORS** for the Screen Controls' Colors list, **PALCOLORS** for the Palette control panel's Colors (Total) list, and **DITHERS** for the Screen Controls' Dither Mode list.

If operation was	RC	ADPRO_RESULT
successful	0	sequence of quoted list entries, with the first entry being the selected item
not successful	10	error string

NOTE ARexx has a limit of 65,535 characters for any string that it sends you. This may affect how much data is received back in **ADPRO_RESULT**.

13.7.12 Select From a List of Items

Syntax: `LISTVIEW title rows [WHAT wtext] [NOSELECT] [SORT]
ITEMS selected_item item1...itemrows`

Display a chooser list with the following attributes:

title

Is the phrase in title bar.

rows

Is the number of entries in the viewable list area.

WHAT *wtext*

Is the **Accept And** *wtext* phrase.

NOSELECT

Specifies that the *selected_item* does not appear in the **ITEMS** list.

SORT

Specifies whether to sort the items or not.

ITEMS *selected_item* *item₁...item_{rows}*

Is the list of items to display in the chooser list; note that *selected_item* should not be specified if **NOSELECT** is given. If you neglect to specify *selected_item* in the list of *items* and you select either **Accept** button, your selection will not be honored and **RC** will return with a value of 5.

If	RC	ADPRO_RESULT
item was "Accept"ed	0	sequence of quoted list entries, with the first entry being the selected item.
"Accept" and ...	1	sequence of quoted list entries, with the first entry being the selected item.
Chooser was "Cancel"led	5	error string
error occurred	10	error string

NOTE ARexx has a limit of 65,535 characters for any string that it sends you. This may affect how much data is received back in ADPRO_RESULT.

13.7.13 Display a Message

Syntax: `DISPLAYMESSAGE` (1)
`DISPLAYMESSAGE message` (2)

The `DISPLAYMESSAGE` command allows you to place a text string in the title bar of the screen on which ADPro is running.

The first form resets the title bar to the default text.

The second form sets the title bar text to the given message. Be sure to enclose the text in a second set of quotes if it has embedded spaces.

Operation	RC	ADPRO_RESULT
always returns	0	undefined

It is the ARexx programmer's responsibility to move ADPro to front (using `ADPRO_TO_FRONT`) before calling this function.

13.7.14 The Notify Requester

Syntax: `OKAY1 "string"`

The `OKAY1` command can be used to display an arbitrary string. The string will be rendered on the same screen as ADPro and execution will pause until the user selects the "resume" button in the `OKAY1` requester.

Operation	RC	ADPRO_RESULT
always returns	1	undefined

It is the ARexx programmer's responsibility to move ADPro to front (using `ADPRO_TO_FRONT`) before calling this function.

13.7.15 The Two-Choice Response Requester

Syntax: `OKAY2 "string"`

The `OKAY2` command can be used to display an arbitrary string and solicit a two valued answer from the user. The string will be rendered on the same screen as ADPro and execution will pause until the user selects the “OK” or “Cancel” buttons in the `OKAY2` requester.

If the user selects	RC	ADPRO_RESULT
“Cancel”	0	undefined
“OK”	non-zero	undefined

It is the ARexx programmer’s responsibility to move ADPro to front (using `ADPRO_TO_FRONT`) before calling this function.

13.7.16 The Multiple-Choice Response Requester

Syntax: `OKAYN 'title' 'text' 'gadget_labels'`

The `OKAYN` command can be used to display an arbitrary string and solicit an *n* valued answer from the user. The *title* argument will be displayed in the requester’s title bar. The *text* argument will be displayed in the “body” or inside the requester. The *gadget_labels* argument describes how many buttons will be on the requester, as well as how each button will be labelled. Each label is separated by a vertical bar (“|”), as in “Yes|No|Cancel”.

Each button has its own button number. The leftmost is 1, the one to the right of that is 2, and so on. However, the rightmost button is always 0.

If the user selects	RC	ADPRO_RESULT
the rightmost button	0	undefined
any other button	button number	undefined

It is the ARexx programmer’s responsibility to move ADPro to front (using `ADPRO_TO_FRONT`) before calling this function.

13.7.17 Loading a Settings File

Syntax: `LOAD_DEFAULTS filename`

This command loads the ADPro defaults file specified by *filename*.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	error message

13.7.18 Saving a Settings File

Syntax: **SAVE_DEFAULTS** *filename*

This command saves as many of the program settings to the default file under the name *filename*.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	error message

13.7.19 Obtaining Exclusive Use of ADPro

Syntax: **OBTAIN_ADPRO**

This command tries to place an exclusive “lock” on ADPro, so that your script will have sole access of the program. This is very important when two or more scripts are likely to call ADPro at the same time.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.7.20 Releasing Exclusive Use of ADPro

Syntax: **RELEASE_ADPRO**

This command will release the exclusive “lock” on ADPro, so that other scripts may be able to lock the program.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.8 Loaders

Loaders can be selected from the main window or through their ARexx interfaces. For specific information, read each loader's control panel and ARexx interface sections.

13.8.1 Setting the Load Format

Syntax: **LFORMAT** (1)
 LFORMAT *format* (2)

The first form of the command will fill the **ADPRO_RESULT** field with the name of the currently selected Load Format, if results are requested.

The second form of the command sets the Load Format to the specified string.

If module	RC	ADPRO_RESULT
exists	0	the <i>previously</i> selected Load Format
does not exist	10	"Unknown Module"

Refer to Section 5.3.1 and Chapter 6 for more information concerning Load Formats.

13.8.2 Invoking the Current Loader

Syntax: **LOAD** *filename* [*loader_opts*] [*comp_opts*] [**FORCE_PALETTE**]
This command causes the specified file to be loaded. The only required argument is the filename to be loaded. If you wish to allow the user to manually enter a filename to load, then you must supply a NULL *filename* (in ARexx, this would be: "' || '").

The loader which will be invoked is specified with the **LFORMAT** command.

The *loader_opts* arguments are loader dependent. The currently defined loader-specific arguments are defined in each loader's ARexx Interface reference section.

comp_opts define the ARexx interface to the image compositing capability of ADPro. See Section 13.8.5 for a list of the currently defined *comp_opts*.

The **FORCE_PALETTE** argument allows the loaded image's palette (if it has one) to replace the current palette, even if the palette is currently locked. The **FORCE_PALETTE** option, if desired, must be the last option specified.

If load was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	error string

13.8.3 Invoking a Loader

Syntax: `LOADER loader_module filename loader_args`

An alternative to the `LFORMAT ... LOAD` sequence is the `LOADER` command. It takes both the name of the loader and its arguments on one line. This is a very useful time-saver in that you can set and call a loader using one command (`LOADER`). Note that you must specify a *filename* as well as a module name (*loader_module*).

13.8.4 Launching a Loader

Syntax: `LOAD_GUI`

This command will launch the currently selected loader as if it were selected manually from ADPro's main window.

Pseudo loader modules **cannot** be executed with this command.

If GUI was	RC	ADPRO_RESULT
was opened	0	undefined
could not be opened	10	"Error"

13.8.5 Compositing Options

The *comp_opts* parameters can take one of two forms: Old Style and New Style arguments. Most loaders have been updated to the New Style, which is more flexible and readable. However, some loaders still use the Old Style. Both methods are described in this section.

Old Style

`left_edge top_edge` (1)
`left_edge top_edge mix_level` (2)
`left_edge top_edge mix_level rt gt bt` (3)

In all three forms, you must specify where the image to be loaded will be placed relative to the upper left-hand corner of the image currently in memory. This is specified by *left_edge* and *top_edge*.

The first form will use a mix level of 100, whereas the other forms require that you specify the mix level to be used (which may range in value from 1 to 100).

The third form allows you to specify which 24 bit-plane color value will be considered transparent. If you are merging a grayscale image, the gray level to be considered transparent is taken to be *rt*. In the first form, transparency is disabled.

New Style

The New Style compositing arguments can be placed anywhere on the argument line after the *filename* argument. Zero or more of the following can be defined:

COMPOFFSET *left top*

Specify the image offset of the foreground image relative to the background image.

GUI Equivalent: **X Offset:** and **Y Offset:** values.

Constraints: None

Required: No; default: *left* = 0; *top* = 0

COMPMIX *mix*

Specify the mix percentage.

GUI Equivalent: **Mix:** value.

Constraints: $0 \leq \text{mix} \leq 100$

Required: No; default: *mix* = 100

COMPTRANS *r g b*

Specify the transparent color (in the foreground image). If specifying a transparent color range, these values specify one end of the range.

GUI Equivalent: **From Red:**, **Green:**, and **Blue:** values.

Constraints: $-1 \leq r \leq 255$; $-1 \leq g \leq 255$; $-1 \leq b \leq 255$

Required: No; default: no transparency
(*r* = -1; *g* = -1; *b* = -1)

COMPTRANSTO *r g b*

Specify the other end of a transparent color range. **COMPTRANS** must also be defined.

GUI Equivalent: **To Red:**, **Green:**, and **Blue:** values.

Constraints: $-1 \leq r \leq 255$; $-1 \leq g \leq 255$; $-1 \leq b \leq 255$

Required: No; default: **COMPTRANS** specification

ASALPHA

Load the image into alpha memory, converting to 8 bit grayscale if necessary.

GUI Equivalent: **As Alpha** button.

Constraints: **COMPOFFSET** must also be specified.

Required: No; default: current load mode

ALPHAMEM

Compose image using the alpha channel currently in alpha memory.

GUI Equivalent: **Alpha**: set to **Alpha Memory**.

Constraints: **COMPOFFSET** must also be specified.

Required: No; default: load into primary buffer

ALPHA_ONLY

Load only the embedded alpha channel of a file, if it has one.

GUI Equivalent: **Alpha Only** checkbox.

Constraints: None

Required: No; default: load primary image data

USE_ALPHA

Compose image using the alpha channel embedded within the incoming file.

GUI Equivalent: **Alpha**: set to **Use Incoming Alpha**.

Constraints: None

Required: No; default: use **COMPTRANS** transparency

NOPAD

Specify that raw data should not be created.

GUI Equivalent: None

Constraints: None

Required: No; default: create raw data

ONLYPAD

Specify that raw data should only be created.

GUI Equivalent: None

Constraints: None

Required: No; default: create both raw and rendered if possible

For more information about image compositing, please refer to Section 5.3.3s-CompositingPanel.

Some examples follow:

LOAD Filename

This simply loads the file specified by the variable **Filename**. This image completely replaces any previous image in memory.

```
LFORMAT "IFF"  
LOAD Filename 0 0 100 255 255 255
```

This shows an example of loading an IFF-ILBM file with the image compositing options present. This will load the image at an offset of 0,0. The mix level is 100 percent and pure white will be considered to be transparent.

```
LOADER "UNIVERSAL" Filename1  
LOADER "UNIVERSAL" Filename2 ASALPHA COMPOFFSET 100 50  
LOADER "UNIVERSAL" Filename3 ALPHAMEM COMPOFFSET 100 50
```

This shows an example of the useful **LOADER** command and alpha channel compositing using alpha memory. The first command loads in the “background” image. The second command loads in another image into alpha memory. The third command loads in our foreground image using the alpha channel currently in alpha memory, positioning it 100 pixels to the left and 50 pixels down from the upper left corner.

NOTE The alpha channel and foreground images **MUST** be the same width and height, although not the same format. Also note that if you specify offsets for the foreground, you **must** also specify the same offsets for the alpha channel. This is very important.

13.8.6 Load Orientation

Syntax: **ORIENTATION** (1)
ORIENTATION (PORTRAIT | LANDSCAPE) (2)

The first form of the command returns the current load orientation. The **ADPRO_RESULT** variable will be set to either **PORTRAIT** or **LANDSCAPE**.

The second form of the command sets the load orientation to the specified value.

NOTE If you set the orientation to **LANDSCAPE**, the load mode (see Section 13.8.7) will be switched to **REPLACE** automatically. This is to reflect the restriction that you cannot compose an image in a landscaped orientation.

For more information about the uses of portrait and landscape orientations, please refer to Section 5.3.2.

13.8.7 Load Mode

Syntax: **LOAD_TYPE** (1)

LOAD_TYPE (REPLACE | COMPOSE) (2)

The first form retrieves the current state of the **Compositing** control ("REPLACE" for **Replace** or "COMPOSE" for **Compose**), storing it in **ADPRO_RESULT**.

The second form sets the **Compositing** control, using the strings described in the first form.

If operation was	RC	ADPRO_RESULT
successful	0	state information
unsuccessful	10	error string

13.8.8 ALPHA

Syntax: **LOADER "ALPHA" *image_to_load* *alpha_channel_file***

alpha_channel_file should be the name of the grayscale IFF-ILBM file to use as the alpha channel.

The *comp_opts* arguments (old format) must follow the *alpha_channel_file* parameter.

Be sure you do not delete the **.ALPHA** and **ALPHA** files from the **Loaders2** directory. The ALPHA loader needs both of these files to operate. These files are placed in this directory by the installation program.

If load was	RC	ADPRO_RESULT
successful (not using NOPAD)	0	undefined
successful (using NOPAD)	5	undefined
unsuccessful (either case)	10	undefined

NOTE You might want to use the new alpha compositing options. See Section 13.8.5 for more information.

13.8.9 ANIM

Syntax: **LOADER "ANIM" *filename* *loader_opts***

The *loader_opts* for the ANIM loader are:

(FRAME *n* | COUNT | QUIT)

Specify whether to load a specific frame (**FRAME**), count how many frames are available, or close the ANIM file so that other programs may use it.

GUI Equivalent: **Frame Number:** for **FRAME** *n*;
 Count Frames for **COUNT**; **Quit** button.

Constraints: $0 \leq n$

Required: Yes

If load was	RC	ADPRO_RESULT
successful (not using NOPAD)	0	undefined
successful (using NOPAD)	5	undefined
unsuccessful (either case)	10	Error message

13.8.10 BACKDROP

Syntax: **LOADER** "BACKDROP" "XXX" *loader_opts*

where *loader_opts* are the following:

WIDTH *w*

Set the width of the backdrop.

GUI Equivalent: **Width:** value.

Constraints: $1 \leq w \leq 16383$

Required: Yes

HEIGHT *h*

Set the height of the backdrop.

GUI Equivalent: **Height:** value.

Constraints: $1 \leq h \leq 16383$

Required: Yes

(**COLOR** | **GRAY**)

Specify the type of backdrop data.

GUI Equivalent: **Type:** setting.

Constraints: None

Required: Yes

(**UL** *r g b* **UR** *r g b* **LL** *r g b* **LR** *r g b* | **FILL** *r g b*)

Specify the mode and colors. For grayscale data, only the green (*g*) value is used.

GUI Equivalent: **Mode:** and color well settings.

Constraints: $0 \leq r \leq 255$;

$0 \leq g \leq 255$;

$0 \leq b \leq 255$

Required: No; default: black fill

NOTE The old ARexx command syntax is still supported. We, however, suggest you create new scripts with this new syntax—if you will only be using these scripts with newer versions of ADPro.

If backdrop was	RC	ADPRO_RESULT
created	0	undefined
not created	non-zero value	undefined

13.8.11 BACKLINE

Syntax: **LOADER "BACKLINE" "XXX" loader_opts**

where *loader_opts* are the following:

WIDTH *w*

Set the width of the backline.

GUI Equivalent: **Width:** value.

Constraints: $1 \leq w \leq 16383$

Required: Yes

HEIGHT *h*

Set the height of the backline.

GUI Equivalent: **Height:** value.

Constraints: $1 \leq h \leq 16383$

Required: Yes

COLOR | GRAY

Specify the type of backline data.

GUI Equivalent: **Type:** setting.

Constraints: No

Required: Yes

POSITION *p*

Specify the middle position color (as a percentage of distance between start and end).

GUI Equivalent: **Middle:** value.

Constraints: $0 \leq p \leq 100$

Required: Yes

(NW | N | NE | W | E | SW | S | SE)

Specify the direction of the backline, from start to end color.

GUI Equivalent: **Direction:** setting.

Constraints: None

Required: Yes

(**START** *r g b* **MIDDLE** *r g b* **END** *r g b* | **GRAYVALS** *g1 g2 g3*)

Set the colors for the start, middle, and end color wells. For grayscale data, use the **GRAYVALS** keyword and follow it with the start, middle, and ending gray levels.

GUI Equivalent: Color well values.

Constraints: $0 \leq r \leq 255$;
 $0 \leq g \leq 255$;
 $0 \leq b \leq 255$;
 $0 \leq g1 \leq 255$;
 $0 \leq g2 \leq 255$;
 $0 \leq g3 \leq 255$

Required: Yes

NOTE The old ARexx command syntax is still supported. We, however, suggest you create new scripts with this new syntax—if you will only be using these scripts with newer versions of ADPro.

If backline was	RC	ADPRO_RESULT
created	0	undefined
not created	non-zero value	undefined

13.8.12 BMP

Syntax: **LOADER** "BMP" *filename loader_opts*

The only BMP loader *loader_opts* is **NOPAD**, which disables the conversion of rendered image data back into 24 or 8 bit-plane raw data during loading.

If load was	RC	ADPRO_RESULT
successful (not using NOPAD)	0	undefined
successful (using NOPAD)	5	undefined
unsuccessful (either case)	10	undefined

13.8.13 CDXL

Syntax: **LOADER** "CDXL" *filename loader_opts*

The *loader_opts* for the CDXL loader are (note that the audio and pad related options are left undocumented here because they would be available to any CDTV/CD32 developer, and is only appropriate for developers):

FRAMENUM *n*

Load a particular frame.

GUI Equivalent: **Frame Number:** selector.

Constraints: $1 \leq n \leq \text{num_frames}$

Required: No

KEEPOPEN

Keep the file open. See the note below.

GUI Equivalent: None

Constraints: None

Required: No

CLOSE

Close the file and do nothing else.

GUI Equivalent: **Cancel** button.

Constraints: None

Required: No

QUIT

Close the file, quit (unresident the loader), and do nothing else.

GUI Equivalent: **Quit** button.

Constraints: None

Required: No

NOTE When you're through with a file which you explicitly asked to "KEEPOPEN", you must remember to issue a "CLOSE", or risk the file still being open when you reboot, with potentially hazardous results.

If load was	RC	ADPRO_RESULT
successful	0	undefined
unsuccessful	10	Error message

13.8.14 CLIPBOARD

Syntax: **LOADER "CLIPBOARD" "XXX" loader_opts**

The CLIPBOARD loader supports the **NOPAD** keyword for *loader_opts*. If the image in the Clipboard contains rendered image data (i.e., neither 24 bit-plane color nor 8 bit-plane gray), the **NOPAD** argument tells ADPro *not* to convert the rendered image data back into raw image data. Because this option disables the conversion on the newly loaded image into 24 bit-plane color or 8 bit-plane grayscale data, the image is loaded more quickly.

If load was	RC	ADPRO_RESULT
successful (not using NOPAD)	0	undefined
successful (using NOPAD)	5	undefined
unsuccessful (either case)	10	undefined

13.8.15 DPAINT

Syntax: **LOADER "DPAINT" "XXX" loader_opts**

The *loader_opts* for the DPAINT loader are:

("PAGE" | "DISPLAYED" | "PALETTE")

Set the image area to load ("PAGE" or "DISPLAYED") or retrieve the current DPaint palette ("PALETTE").

GUI Equivalent: **Area To Load:** cycle gadget setting.

Constraints: None

Required: Yes

("SHOW" | "HIDE")

Select whether to show the DPaint IV AGA screen or not.

GUI Equivalent: **Show DPaint** button.

Constraints: None

Required: No; default: "HIDE"

"FRAME" *n*

Select the DPaint image buffer to load. Frame 0 is the swap buffer, while numbers greater than 0 specify particular ANIM frames; a single image is equivalent to a one frame ANIM.

GUI Equivalent: **Frame Number:** slider value.

Constraints: $0 \leq n \leq (\text{number_of_ANIM_frames} + 1)$

Required: No; default: $n = 0$

STATUS

Return (in **ADPRO_RESULT**) various information about DPaint. This information is a string of words and numbers of the form:

PW *page_width* PH *page_height* PD
page_depth SW *screen_width* SH *screen_height* LE *left_edge* TE
top_edge DW *display_width* DH *display_height* CP *current_page* NP
number_of_pages VER *DPaint_version* REV *DPaint_revision*

GUI Equivalent: None

Constraints: None

Required: No; default: no status information returned.

If load was	RC	ADPRO_RESULT
successful (not using NOPAD)	0	undefined
successful (using NOPAD)	5	undefined
unsuccessful (either case)	10	Error message

13.8.16 DPIIE

Use the IFF loader to load DPIIE files.

NOTE ARexx scripts that reference the DPIIE loader will have to be modified to use IFF instead.

13.8.17 DV21

Syntax: **LOADER "DV21" filename**

No loader options are used by the DV21 loader.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.8.18 FC24

Syntax: **LOADER "FC24" "XXX" loader_opts**

where *loader_opts* can be the following:

BOARD_NO *num*

Set the board whose buffer will be read.

GUI Equivalent: **Board** cycle gadget setting.

Constraints: $0 \leq num \leq 6$

Required: No; default: current board

GRAB_FROM (A | B)

Set the board buffer to grab.

GUI Equivalent: **Buffer** cycle gadget setting.

Constraints: None

Required: No; default: current buffer

B_WIDTH (384 | 512 | 768 | 1024)

Set the width of the board.

GUI Equivalent: **Width** cycle gadget setting.

Constraints: None

Required: No; default: current width

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.8.19 FLC

Syntax: **LOADER "FLC" filename loader_opts**

The FLC loader can take the following *loader_opts* option:

(*n* | COUNT)

Load a particular frame (*n*) or return the number of frames (in ADPRO_RESULT) in the file

GUI Equivalent: .

Constraints: **Frame:** value.

Required: $1 \leq n \leq \text{num_frames}$

Yes

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	error message

13.8.20 FRACTAL2000

Syntax: **LOADER "FRACTAL2000"** *filename*

The FRACTAL2000 loader does not take any *loader_opts*. The settings that it uses are the ones internal to the selected file.

If load was	RC	ADPRO_RESULT
successful	0	undefined
unsuccessful	10	Error message

13.8.21 FRAMEGRABBER

Syntax: **LOADER "FRAMEGRABBER" "XXX"** *type*

where *type* can be one of:

FIELD1

This causes the loader to download only the first field grabbed.

This results in an image which measures 320 by 200 pixels in size.

FIELD2

This causes the loader to download only the second field grabbed.

This results in an image which measures 320 by 200 pixels in size.

STACKED

This causes the loader to download both of the grabbed fields.

Field 1 will be stacked above field 2. This option results in an image which measures 320 by 400 pixels in size.

INTERLACED

This causes the loader to download both of the grabbed fields which are interlaced together to form one image. This option results in an image which measures 320 by 400 pixels in size.

If no grab type is specified (allowable only if you are not specifying any compositing options, then **INTERLACED** is assumed.

NOTE If you intend to specify compositing options, then you **must** specify a grab type!

It is not reasonable to expect to use the FRAMEGRABBER loader from ARexx to grab frames in a timing dependent fashion because there is no precise way of predicting exactly when the loader will actually grab the frame. The intended method for using this loader from ARexx is to set up the image to be grabbed (perhaps by issuing commands to a VCR or VideoDisc player), allow time for the image to become stable, and then execute the FRAMEGRABBER loader.

NOTE The FRAMEGRABBER loader does not operate correctly on PAL machines.

If load was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.8.22 FRAMESTORE

Syntax: **LOADER "FRAMESTORE"** *filename*

There are no *loader_opts* for the FRAMESTORE loader.

The *comp_opts* arguments (new format) must follow the *filename* parameter. It uses the New Style of compositing arguments.

If load was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.8.23 GIF

Syntax: **LOADER "GIF"** *filename*

No loader options are used by the GIF loader.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.8.24 HAME

Syntax: **LOADER "HAME" filename**

No loader options are used by the HAME loader.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.8.25 ICO

Syntax: **LOADER "ICO" filename**

The ICO loader simply requires a filename to be specified.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.8.26 ICON

The ICON loader can take the following *loader_opts*:

SELECTED

Load the selected icon imagery.

GUI Equivalent: **Selected** option.

Constraints: None

Required: No; default: load the normal image

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.8.27 IFF

Syntax: **LOADER "IFF" filename loader_opts**

The IFF loader supports the **NOPAD** keyword for *loader_opts*. If the IFF-ILBM file being read contains rendered image data (i.e., neither 24 bit-plane color nor 8 bit-plane gray), the **NOPAD** argument tells ADPro *not* to convert the rendered image data back into raw image data. Because this option disables the conversion on the newly loaded image into 24 bit-plane color or 8 bit-plane grayscale data, the image is loaded more quickly. This might be used from

ARexx to implement a simple “slide-show” type program. This feature also makes conversions between same depth file formats much faster.

If load was	RC	ADPRO_RESULT
successful (not using NOPAD)	0	undefined
successful (using NOPAD)	5	undefined
unsuccessful (either case)	10	undefined

Note that when `ADPRO_RESULT = 5`, it means that something was loaded, but not converted into raw image data. Therefore you shouldn't check `RC` this way:

```
/* WRONG WAY */
LOADER "IFF" "SOMETHING" NOPAD
IF (RC ~= 0) THEN DO
    OKAY1 "FAILED!"
    EXIT 10
END
```

but rather:

```
/* RIGHT WAY */
LOADER "IFF" "SOMETHING" NOPAD
IF (RC = 10) THEN DO
    OKAY1 "FAILED!"
    EXIT 10
END
```

13.8.28 IMPULSE

Syntax: `LOADER "IMPULSE" filename`

No loader options are used by the IMPULSE loader.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.8.29 IV24

Syntax: `LOADER "IV24" "XXX" loader_opts`

The IV24 loader's *loader_opts* must conform to one the following:

- no options specified (1)
- width height* (2)
- width height xoff yoff* (3)

In the first form, the width, height, and offsets are taken from default values.

In the second form, the width and height are specified, while the offsets are taken from defaults.

In the third form, the width, height, and offsets are all specified.

An error is returned if the *width* plus the x offset (*xoff*) is larger than 768. An error is also returned if the *height* plus the y offset (*yoff*) is larger than 566. In both cases, the error is specified by an RC value of 10. **ADPRO_RESULT** is undefined after an error.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.8.30 JPEG

Syntax: **LOADER "JPEG" filename loader_opts**

The only JPEG loader *loader_opts* option is:

SMOOTHING

Cause a smoothing operation to be performed while the image is being decompressed.

GUI Equivalent: **Smoothing:** cycle gadget set to **ON**.

Constraints: None

Required: No; default: no smoothing

If load was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.8.31 MACPAINT

Syntax: **LOADER "MACPAINT" filename**

No loader options are used by the MACPAINT loader.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.8.32 PAR_PEG

Syntax: **LOADER "PAR_PEG" filename loader_opts**

The PAR_PEG loader takes the following arguments:

(**FRAME** *n* | **COUNT**)

Specify a frame to load or count the number of frames in the specified file. For **COUNT**, the number of frames is returned in **ADPRO_RESULT**. Still images will return 0.

GUI Equivalent: **Frame:** value.

Constraints: $1 \leq n \leq \text{num_frames}$

Required: No; default: load still image

(**FIELD1** | **FIELD2**)

Specify the field to load.

GUI Equivalent: **Mode:** set to one of the field choices.

Constraints: None

Required: No; default: load a full frame

DOUBLE

When loading a field, double each scan line so that the resulting image fills a frame. This argument is ignored if neither **FIELD1** nor **FIELD2** is defined.

GUI Equivalent: **Mode:** set to one of the double-field choices.

Constraints: None

Required: No; default: do not double lines

If load was	RC	ADPRO_RESULT
successful	0	undefined
unsuccessful	10	Error message

13.8.33 PCX

Syntax: **LOADER "PCX" filename loader_opts**

The PCX loader supports the **NOPAD** keyword for *loader_opts*. If the PCX file being read contains rendered image data (i.e., neither 24 bit-plane color nor 8 bit-plane gray), the **NOPAD** argument tells ADPro *not* to convert the rendered image data back into raw image data. Because this option disables the conversion on the newly loaded image into 24 bit-plane color or 8 bit-plane grayscale data, the image is loaded more quickly. This might be used from ARexx to implement a simple "slide-show" type program. This feature also makes conversions between same depth file formats much faster.

If load was	RC	ADPRO_RESULT
successful (not using NOPAD)	0	undefined
successful (using NOPAD)	5	undefined
unsuccessful (either case)	10	undefined

13.8.34 POINTER

Syntax: **LOADER "POINTER" "XXX"**

The **POINTER** loader can be called from **ARexx** and will *pause the ARexx program until the correct hot-key sequence is depressed on the keyboard*. The correct sequence is **Right Alt + Right Shift + Backspace**.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.8.35 QRT

Syntax: **LOADER "QRT" filename**

No loader options are used by the **QRT** loader.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.8.36 SCREEN

Syntax: **LOADER "SCREEN" screen_name**

The *screen_name* argument specifies which screen is to be grabbed. The specified string is used as a prefix when matching against the Intuition names of each defined screen. By allowing the name you supply to be used as a prefix, the **SCREEN** loader can easily grab screens with very long screen names or which have screen names which change over time.

For example, ASDG's high performance editor, CygnusEd Professional (CED-Pro), has a very long Intuition screen name which includes status information that changes over time. CEDPro screens always start with the characters, **CED**. Therefore, specifying only **CED** is enough to cause a CEDPro screen to be grabbed.

Specifying a **NULL** ("") file name causes the frontmost screen to be grabbed. In this way, screens with no Intuition screen name can be grabbed. The **ARexx**

program would first have to arrange for the nameless screen to be popped-to-front.

Specifying a *screen_name* of "?" does not cause a screen to be grabbed. Rather, it returns a list of the screens (having names) which are currently defined. This list will be returned in the **ADPRO_RESULT** variable. Each screen in the list will be surrounded by a double quote.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.8.37 SCULPT

Syntax: **LOADER "SCULPT"** *filename loader_opts*

The SCULPT loader's *loader_opts* must conform to the following:

width height type

where:

width

Is the width of the image to be loaded.

height

Is the height of the image to be loaded.

type

Can be either "COLOR" or "GRAY".

These arguments must all be present. Composition options may follow these arguments.

NOTE From ARexx, you may specify only one file on the **LOAD** statement. If loading a color image, this file name must correspond to the red component, which must end in the three letters, "red". In this case, the files to be loaded must end in the conventional extensions. When loading a grayscale image, you must completely specify the name of the single file to be loaded.

If load was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.8.38 TEMP

Syntax: `LOADER "TEMP" "XXX"`

There are no *loader_opts* for the TEMP loader.

The *comp_opts* arguments (new format) must follow the "XXX" parameter.

If load was	RC	ADPRO_RESULT
successful (not using NOPAD)	0	undefined
successful (using NOPAD)	5	undefined
unsuccessful (either case)	10	undefined

13.8.39 UNIVERSAL

Syntax: `LOADER "UNIVERSAL" filename`

The UNIVERSAL loader requires some special discussion. You can use the UNIVERSAL loader from ARexx. However, since you don't know which loader will ultimately be used to load the image, you cannot make use of any loader options.

If load was	RC	ADPRO_RESULT
successful (not using NOPAD)	0	undefined
successful (using NOPAD)	5	undefined
unsuccessful (either case)	10	undefined

13.8.40 VLAB

Syntax: `LOADER "VLAB" "XXX"`

No loader-specific options (*loader_opts*) are available. The loader will use the the current settings from the control panel.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.8.41 YUVN

Syntax: `LOADER "YUVN" filename`

No loader-specific options (*loader_opts*) are available.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.9 Savers

Savers can be selected from the main window or through their ARexx interfaces. For specific information, read each saver's control panel and ARexx interface sections.

13.9.1 Setting the Save Format

Syntax: **SFORMAT** (1)
SFORMAT *format* (2)

The first form of the command fills the **ADPRO_RESULT** field with the name of the currently selected Save Format, if results are requested.

The second form of the command sets the Save Format to the specified string.

If module	RC	ADPRO_RESULT
exists	0	the <i>previously</i> selected Save Format
does not exist	10	"Unknown Module"

Refer to Section 5.4.1 and Chapter 7 for more information concerning Save Formats.

13.9.2 Invoking the Current Saver

Syntax: **SAVE** *filename type saver_opts*

This command invokes a Saver with the specified arguments.

The Saver which will be invoked is specified with the **SFORMAT** command.

A filename and save type are required arguments. The type must be one of the following strings: **RAW**, **IMAGE**, or **SCREEN**. Specifying **RAW** will attempt to save the raw image data, if available. Specifying **IMAGE** will attempt to save the rendered image data, if available.

Specifying **SCREEN** will attempt to save only as much rendered image data as will fit in the currently defined screen type (and, will save only a screen sized portion of the image if it is larger than the current screen type). Screen size is defined by the currently set screen width and height. Also, the upper left hand corner of the portion of the image to be saved is determined by scrolling the image by hand. From ARexx, you cannot automatically offset the portion of the image to be saved.

Each particular saver may have its own restrictions as to which type of data may be saved. These restrictions are defined in each saver's ARexx Interface reference section.

Remember that if you only have rendered image data, you can turn it back into raw image data by saving the rendered image data to disk and then reloading it.

The *saver_opts* are saver specific options. The currently defined saver-specific arguments are defined in each saver's ARexx Interface reference section.

13.9.3 Invoking a Saver

Syntax: **SAVER** *saver_module filename saver_args*

An alternative to the **SFORMAT ... SAVE** sequence is the **SAVER** command. It takes both the name of the saver and its arguments on one line. This is a very useful time-saver in that you can set and call a saver using one command (**SAVER**). Note that you must specify a *filename* as well as a module name (*saver_module*).

13.9.4 Launching a Saver

Syntax: **SAVE_GUI**

This command will launch the currently selected saver as if it were selected manually from ADPro's main window.

Pseudo saver modules **cannot** be executed with this command.

If GUI was	RC	ADPRO_RESULT
was opened	0	undefined
could not be opened	10	"Error"

13.9.5 A2410

Syntax: **SAVER** "A2410" "XXX" *type saver_opts*

The *type* argument must be "IMAGE" only, or "RAW" for 8-bit grayscale images.

The ARexx support for Commodore's A2410 display board constitutes a language in-and-of-itself. Each call to the A2410 saver from ARexx defines a stream of commands (defined as *saver_opts*) which are parsed and executed from left to right. These commands are:

BOARD_NO *num*

If you have more than one A2410 installed, you can use this command to select which board subsequent commands will be sent to.

All commands described here operate on the currently selected board.

B_SIZE *id*

Resolution	Interlaced	ID
1024 x 768	No	0
800 x 600	No	1
640 x 400	Yes	2
1024 x 768	Yes	3
1024 x 1024	No	4
736 x 480	No	5
1024 x 1024	No	6

Table 13.1: TIGA screen IDs which may be used as arguments to the B_SIZE parameter.

This command sets display resolution to match the TIGA screen *id* as given in Table 13.1. If the crystal matching the specified *id* is unavailable (i.e., not installed on the board), then the command will fail.

CENTER

This command sets the display offsets to center the image currently in Amiga memory on the currently selected A2410 display.

CLEAR

This command clears the currently selected board.

IMAGE

This command causes the image currently in Amiga memory to be downloaded to the currently selected board at the current offsets.

SET_DOX *left_edge*

This command specifies which column (DOX means "Destination Offset X") the image will be written to.

SET_DOY *top_edge*

This command specifies which row (DOY means "Destination Offset Y") the image will be written to.

SET_DWX *width*

This command sets the width of the image area to be written to the specified value. The required argument *width* is in A2410 pixels.

SET_DWY *height*

This command sets the height of the image area to be written to the specified value. The required argument *height* is in A2410 pixels.

SET_SOX *left_edge*

This command specifies the left edge of the image area *within the image* in Amiga memory which will be written to the board (SOX means "Source Offset X").

SET_SOY *top_edge*

This command specifies the top edge of the image area *within the image* in Amiga memory which will be written to the board (**SOY** means "Source Offset Y").

Since ARexx command lines invoking the A2410 saver can be complex, finding an error on the command line can become difficult. The saver, therefore, returns in **ADPRO_RESULT** the number of the argument in which an error was found. The argument numbering begins with 1, which corresponds to the *type* parameter of the ARexx **SAVER** command. If an error is encountered, **RC** will return non-zero.

For instance, the following returns a 1 in **ADPRO_RESULT**:

```
SAVER "A2410" "XXX" "NEITHER_IMAGE_NOR_SCREEN" "IMAGE"
```

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	parameter error number	undefined

13.9.6 ANIM

Syntax: **SAVER "ANIM" filename saver_opts**

The ANIM saver may be instructed to save either **IMAGE** or **SCREEN** data (specified as one of the *saver_opts*).

The ANIM saver's *saver_opts* include:

(IMAGE | SCREEN)

Set the type of rendered data to save.

GUI Equivalent: **Save Type** cycle gadget value.

Constraints: None

Required: Yes

(APPEND | WRAPUP | QUIT)

Describe the operation to perform. **APPEND** adds the current rendered data to the end of the ANIM. **WRAPUP** appends the first two frames to the end of the ANIM *and* closes the file. **QUIT** closes the file so that other programs can access it.

GUI Equivalent: **Append, Wrap Up, and Quit** buttons on the control panel.

Constraints: None

Required: Yes

TRUNCATE *n*

TRUNCATE (takes an integer argument) either reduces the file to *n* frames (for positive *n*) or chops $-n$ frames (for negative *n*).

GUI Equivalent: **Truncate** button on the control panel. The *n* value is equivalent to the value specified in the Truncate control panel's input field.

Constraints: None

Required: No; default: no truncation

(FASTER | SMALLER)

Set the compression level or quality. **FASTER** encodes the file using the old technique, whereas **SMALLER** creates smaller ANIM files but takes a little longer to create them. Playback speed should be equivalent.

GUI Equivalent: **Compression Quality** setting.

Constraints: None

Required: No; default: **FASTER**

(BYTE | WORD | LONG)

Set the compression mode. **BYTE** is the traditional ANIM OpCode-5 mode, whereas **WORD** and **LONG** is ANIM OpCode-8.

GUI Equivalent: **Compression Type** setting.

Constraints: None

Required: No; default: current setting; **BYTE** if first time.

If save was	RC	ADPRO_RESULT
successful	0	undefined
unsuccessful	10	Error message

13.9.7 BMP

Syntax: **SAVER "BMP" filename type**

The *type* argument must be either **"RAW"** or **"IMAGE"**.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.9.8 CDXL

Syntax: **SAVER "CDXL" filename type saver_opts**

For *type*, the CDXL saver may be instructed to save either **RAW** or **IMAGE** data.

The *saver_opts* can be zero or more of the following (note that the audio and pad related options are left undocumented here because they would be available to any CDTV/CD32 developer, and is only appropriate for developers):

FRAMENUM *n*

Save to particular frame number. If “-1”, append frame. Otherwise, replace frame *n*.

GUI Equivalent: **Frame Number:** selector.

Constraints: -1;
or $1 \leq n \leq \text{num_frames} + 1$

Required: No

KEEPOPEN

Keep the file open. See the note below.

GUI Equivalent: None

Constraints: None

Required: No

CLOSE

Close the file and do nothing else.

GUI Equivalent: **Cancel** button.

Constraints: None

Required: No

QUIT

Close the file, quit (make unresident) the saver, and do nothing else.

GUI Equivalent: **Quit** button.

Constraints: None

Required: No

INTERLEAVED

Use interleaved format.

GUI Equivalent: **Image Type: Interleaved**

Constraints: None

Required: No; default: save non-interleaved

NOTE When you're through with a file which you explicitly asked to “KEEPOPEN”, you must remember to issue a “CLOSE”, or risk the file still being open when you reboot, with potentially hazardous results.

If save was	RC	ADPRO_RESULT
successful	0	undefined
unsuccessful	10	Error message

13.9.9 CLIPBOARD

Syntax: **SAVER** "CLIPBOARD" *filename type*

The *type* argument must be "RAW", "IMAGE", or "SCREEN".

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.9.10 DPAINT

Syntax: **SAVER** "DPAINT" "XXX" *saver_opts*

The *saver_opts* for the DPAINT saver are:

("IMAGE" | "SCREEN")

Set the type of image data to save.

GUI Equivalent: **Image Type:** cycle gadget setting.

Constraints: None

Required: Yes

("FRAME" *n* | "APPEND" | "INSERT" *r* | "DELETE" *r*)

Select where to save the frame (replace "FRAME" *n* or "APPEND" to end of ANIM) or modify an existing range of ANIM frames ("INSERT" *r* or "DELETE" *r*), starting at the current frame position.

GUI Equivalent: **Frame Number:** slider value ("FRAME" *n*);
Append button ("APPEND");
Insert Frame and **Delete Frame** buttons
("INSERT" *r* and "DELETE" *r*)

Constraints: $0 \leq n \leq (\text{number_of_ANIM_frames} + 1)$;
 $r \leq (\text{number_of_ANIM_frames} - \text{current_frame})$

Required: Yes

("PRESERVE" | "CHANGE")

Select whether to preserve or change the DPaint palette to the palette of the image to save. Note, for example, that if you are trying to "PRESERVE" a HAM image but DPaint currently has a 8 color palette, this command will fail.

GUI Equivalent: **Screen Format:** cycle gadget setting.

Constraints: None

Required: No; default: "PRESERVE"

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	Error message

13.9.11 DPIIE

Syntax: **SAVER "DPIIE" *filename type***

The *type* argument must be "IMAGE" only.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.9.12 FC24

Syntax: **SAVER "FC24" "XXX" *type saver_opts***

The *type* argument must be either "RAW" or "IMAGE".

The ARexx support for Impulse's FireCracker 24 display board constitutes a language in-and-of-itself. Each call to the FC24 saver from ARexx defines a stream of commands (defined as *saver_opts*) which are parsed and executed from left to right. These commands are:

AMIGA (ON | OFF)

This command turns the Amiga display on or off as specified. This is useful when the Amiga and the FireCracker 24 are sharing the same monitor.

BOARD (ON | OFF)

This command turns the currently selected FireCracker 24 display on or off as specified. This is useful when the Amiga and the FireCracker 24 are sharing the same monitor.

BOARD_NO *num*

If you have more than one FireCracker 24 installed, you can use this command to select which board subsequent commands will be sent to. All commands described here operate on the currently selected board.

B_WIDTH (384 | 512 | 768 | 1024)

This command sets display width to the specified value. Note two things: First, that changing the board width while the board is "on" necessarily causes that board's display to change. Second, the FireCracker 24's double buffering features can be used only when the board width is set to 384 or 512.

CENTER

This command sets the display offsets to center the image currently in Amiga memory.

CLEAR

This command clears the currently selected board or, if the display width is 512 or less, clears the currently selected buffer (as defined by **IMAGE_TO**).

DISPLAY_FROM (A | B)

If the currently selected board's width is 512 or less, then you can take advantage of the FireCracker 24's double buffering capability. This command specifies which of the two buffers will be displayed from and causes the switch to take place immediately.

IMAGE

This command causes the image currently in Amiga memory to be downloaded to the currently selected board and buffer at the current offsets.

IMAGE_TO (A | B)

When double buffering is possible, this command selects which buffer will be written to by a subsequent **IMAGE** command.

SET_DOX *left_edge*

This command specifies which column (DOX means "Destination Offset X") the image will be written to.

SET_DOY *top_edge*

This command specifies which row (DOY means "Destination Offset Y") the image will be written to.

SET_DWX *width*

This command sets the width of the image area to be written to the specified value. The required argument *width* is in FireCracker 24 pixels.

SET_DWY *height*

This command sets the height of the image area to be written to the specified value. The required argument *height* is in FireCracker 24 pixels.

SET_SOX *left_edge*

This command specifies the left edge of the image area *within the image* in Amiga memory which will be written to the board (SOX means "Source Offset X").

SET_SOY *top_edge*

This command specifies the top edge of the image area *within the image* in Amiga memory which will be written to the board (SOY means "Source Offset Y").

Since ARexx command lines invoking the FireCracker 24 saver can be complex, finding an error on the command line can become difficult. The saver, therefore, returns in **ADPRO_RESULT** the number of the argument in which an error was found. The argument numbering begins with 1, which corresponds to the *type* argument of the ARexx **SAVER** command. If an error is encountered, **RC** will return non-zero.

For instance, the following returns a 1 in **ADPRO_RESULT**:

```
SAVER "FC24" "XXX" "NEITHER_RAW_NOR_IMAGE" "IMAGE"
```

If you request a data type (either **RAW** or **IMAGE**) which is not available, **RC** will be set to 10, and **ADPRO_RESULT** will be set to 1.

For an example of how the FireCracker 24 saver is used from ARexx, please see **Example FG and FC24 Use** in Appendix B.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	1

13.9.13 FRAMEBUFFER

NOTE The FRAMEBUFFER saver does not come packaged in this version of Art Department Professional. The saver's ARexx interface is documented for users who have this module from a previous version of Art Department Professional.

Syntax: **SAVER "FRAMEBUFFER" "XXX" type**
The *type* argument must be "RAW" only.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.9.14 FRAMESTORE

Syntax: **SAVER "FRAMESTORE" filename "RAW" saver_opts**
where *saver_opts* can be the following:

- FILTER** *f*
Set the amount of filter that we be done to the saved Framestore.
GUI Equivalent: **Filter:** rocker switch setting.
Constraints: $0 \leq f \leq 4$
Required: No; default: $f = 2$.
- COMPRESS** (0 | 1)
Specify whether to save in compressed Framestore format or not.
GUI Equivalent: **Compression:** rocker switch setting.
Constraints: None
Required: No; default: compressed (1).

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.9.15 GIF

Syntax: **SAVER "GIF" *filename type***

The *type* argument must be "SCREEN" or "IMAGE".

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.9.16 HAME

Syntax: **SAVER "HAME" *saver_opts***

The HAME saver's *saver_opts* must conform to one of the following:

- filename* "IMAGE" "TODISK" (1)
- filename* "IMAGE" "TODISK" *screen_type* (2)
- "XXX" "IMAGE" "DISPLAY" (3)
- "XXX" "IMAGE" "DISPLAY" *ticks* (4)
- "XXX" "IMAGE" "DISPLAY" *ticks screen_type* (5)

The HAME saver can only save **IMAGE** data.

The first form saves the currently rendered HAME data to the file specified by *filename*. If *filename* is NULL, then the file requester will pop up to allow the user to manually select the name of the file in which to save the data. The screen settings used will be what ever had been last used.

The second form saves the currently rendered HAME data to the file specified by *filename*. If *filename* is NULL, then the file requester will pop up to allow the user to manually select the name of the file in which to save the data. The number specified by *screen_type* is used to determine what type of screen to construct. This number can be assembled using the values contained in Table 13.4. Note that the VGA and Super Hi-Res modes are not allowed.

The third form allows you to display the image data, rather than save it to disk. If the "DISPLAY" keyword is present, then the supplied filename is ignored. The image is displayed until the user clicks the left mouse button. The screen settings used to construct the display will be the same ones last used.

The fourth form will display the image data, but for a specific amount of time. In this form, you can specify the number of *ticks* (fiftieths of a second) to hold the image on-screen. The value specified by *ticks* may be either 0 or between 1 and 65535. If the value is 0, then the displayed image will remain on-screen until the left mouse button is clicked.

The fifth form will display the image data on a specific type of screen for a specific amount of time. The *ticks* argument is the same value as described in the fourth form. The *screen_type* argument is the same as described in the second form.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

For more information about preparing rendered image data for use with the HAME, please see Section 7.12.

13.9.17 HARLEQUIN

Syntax: **SAVER "HARLEQUIN" "XXX" *type saver_opts***

The *type* argument must be either "RAW" or "IMAGE".

The ARexx support for ACS's Harlequin is very similar to the support for the Impulse FireCracker 24. Each call to the HARLEQUIN saver from ARexx defines a stream of commands (defined as *saver_opts*) which are parsed and executed from left to right. These commands are:

ALPHA (ON | OFF)

This command turns the alpha channel of the currently selected screen on or off. If the currently selected board is not equipped with alpha channel support, an error will be returned.

BOARD_NO *num*

If you have more than one Harlequin installed, you can use this command to select which board subsequent commands will be sent to. All commands described here operate on the currently selected board.

CENTER

This command sets the display offsets to center the image currently in Amiga memory.

CLEAR

This command clears the currently selected screen.

DISPLAY (0 | 1)

This command is used to select which screen on the currently selected board will be displayed. If the currently selected board does not support double buffering, an error will be returned if you reference a screen which is not available.

GENLOCK (ON | OFF)

This command turns the genlock of the currently selected board on or off as specified. If the currently selected board supports double buffering, setting the genlock status for one screen affects the other screen as well.

IMAGE

This command causes the image currently in Amiga memory to be downloaded to the currently selected board and buffer at the current offsets.

IMAGE_TO (0 | 1)

This command is used to select which screen on the currently selected board will be the destination of all image data sent from the Amiga. If the currently selected board does not support double buffering, referencing the second buffer will generate an error.

LACE (ON | OFF)

This command turns the interlace mode of the currently selected screen on or off as specified. If the currently selected board supports double buffering, setting the interlace status of one screen does not affect the other.

SCREEN (ON | OFF)

This command turns the currently selected screen on or off as specified.

SET_DOX *left_edge*

This command specifies which column (DOX means "Destination Offset X") the image will be written to.

SET_DOY *top_edge*

This command specifies which row (DOY means "Destination Offset Y") the image will be written to.

SET_DWX *width*

This command sets the width of the image area to be written to the specified value. The required argument *width* is in Harlequin pixels.

SET_DWY *height*

This command sets the height of the image area to be written to the specified value. The required argument *height* is in Harlequin pixels.

SET_SOX *left_edge*

This command specifies the left edge of the image area *within the image* in Amiga memory which will be written to the board (SOX means "Source Offset X").

SET_SOY *top_edge*

This command specifies the top edge of the image area *within the image* in Amiga memory which will be written to the board (SOY means "Source Offset Y").

WIDTH (740 | 832 | 910)

This command sets display width of the currently selected board to the specified value. This command changes the width of both buffers if the currently selected board supports double buffering.

Since ARexx command lines invoking the Harlequin saver can be complex, finding an error on the command line can become difficult. The saver, therefore, returns in **ADPRO_RESULT** the number of the argument in which an error was found. The argument numbering begins with 1, which corresponds to the *type* field of the ARexx **SAVER** command. If an error is encountered, **RC** will return non-zero.

For instance, the following returns a 1 in **ADPRO_RESULT**:

```
SAVER "HARLEQUIN" "XXX" "NEITHER_RAW_NOR_IMAGE" "IMAGE"
```

If you request a data type (either **RAW** or **IMAGE**) which is not available, **RC** will be set to 10 and **ADPRO_RESULT** will be set to 1.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	1

13.9.18 ICON

Syntax: **SAVER "ICON" filename saver_opts**

The **ICON** saver's *saver_opts* include:

(**DISK** | **DRAWER** | **TOOL** | **PROJECT** | **TRASHCAN**)

Set the type of icon to save.

GUI Equivalent: **Icon Type to Create** choice.

Constraints: None

Required: Yes

Note that you do not need to specify the ".info" extension—it will be added for you.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.9.19 IFF

Syntax: **SAVER "IFF" filename type**

The IFF saver may be instructed to save **RAW**, **IMAGE**, or **SCREEN** data *type*.

The IFF saver does not have any *saver_opts*.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.9.20 IMPULSE

Syntax: **SAVER "IMPULSE" filename "RAW" format**

The *format* argument must be either **"RGBN"** or **"RGB8"**.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.9.21 IV24

Syntax: **SAVER "IV24" "XXX" type saver_opts**

The *type* argument must be **"RAW"** only.

The ARexx support for GVP's IV24 display board constitutes a language in-and-of-itself. Each call to the IV24 saver from ARexx defines a stream of commands (defined as *saver_opts*) which are parsed and executed from left to right. These commands are:

CENTER

This command sets the display offsets to center the image currently in Amiga memory.

IMAGE *n*

This command causes the image currently in Amiga memory to be downloaded to the IV24. The resulting IV24 display is shown for *n* ticks (fiftieths of a second). If *n* is 0, then the IV24 display is shown until the user clicks the left mouse button. A value for *n* must be supplied.

SET_DOX *left_edge*

This command specifies which column (DOX means "Destination Offset X") the image will be written to.

SET_DOY *top_edge*

This command specifies which row (DOY means "Destination Offset Y") the image will be written to.

SET_DWX *width*

This command sets the width of the image area to be written to the specified value. The required argument *width* is in IV24 pixels.

SET_DWY *height*

This command sets the height of the image area to be written to the specified value. The required argument *height* is in IV24 pixels.

SET_SOX *left_edge*

This command specifies the left edge of the image area *within the image* in Amiga memory which will be written to the board (**SOX** means "Source Offset X").

SET_SOY *top_edge*

This command specifies the top edge of the image area *within the image* in Amiga memory which will be written to the board (**SOY** means "Source Offset Y").

SCREEN_TYPE *stype*

The number specified by *stype* is used to determine what type of screen to construct. This number can be assembled using the values contained in Table 13.4. Note that the VGA, Super Hi-Res, Super72, and Default modes are not allowed.

Since ARexx command lines invoking the IV24 saver can be complex, finding an error on the command line can become difficult. The saver, therefore, returns in **ADPRO_RESULT** the number of the argument in which an error was found. The argument numbering begins with 1, which corresponds to the *type* argument of the ARexx **SAVER** command. If an error is encountered, RC will return non-zero.

For instance, the following returns a 1 in **ADPRO_RESULT**:

```
SAVER "IV24" "XXX" "NOT_RAW" "CENTER" "IMAGE"
```

If raw image data is not available, RC will be set to 10 and **ADPRO_RESULT** will be set to 1.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	1

13.9.22 JPEG

Syntax: **SAVER** "JPEG" *filename* "RAW" *saver_opts* (1)

The JPEG saver may be instructed to save **RAW** data only; this keyword should be used as the value for the type argument (as shown above).

Mode	Value
Hi Resolution On	1
Interlace On	2
Horizontal Overscan On	8
Vertical Overscan On	16

Table 13.2: SCREEN_TYPE mask values for the OPALVISION saver.

The *saver_opts* can be the following:

QUALITY *q*

Set the level of quality.

GUI Equivalent: **Quality:** value.

Constraints: $1 \leq q \leq 1000$

Required: No; default: 32

TABLES *filename*

Specify a custom quantization table to use. The file that is specified must be in IFF DBJS (JStream) format, as generated by Digital Micronics, Inc.'s Digital Broadcaster 16TM and Digital Broadcaster 32TM software.

GUI Equivalent: None

Constraints: None

Required: No; default: use built-in table.

If save was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	Error message

NOTE The old ARexx command syntax is still supported. However, we suggest you write new ARexx scripts with the new syntax.

13.9.23 OPALVISION

Syntax: **SAVER "OPALVISION" "XXX" type saver_opts**

The *type* argument must be one of "RAW", "IMAGE", or "SCREEN".

The *saver_opts* for the OPALVISION saver are:

SCREEN_TYPE *stype*

The display resolution, similar to ADPro's **Screen Controls** ARexx interface. Note, however, that only the flags specified in Table 13.2 are allowed.

GUI Equivalent: Resolution control rocker switch.

Constraints: See table above.

Required: No; default: current setting

DURATION *t*

Amount of time in ticks (fiftieths of a second) to display the image.

GUI Equivalent: None

Constraints: $0 \leq t$

Required: No; default: 0

A couple of examples are:

```
SAVER "OPALVISION" "XXX" "RAW" SCREEN_TYPE 3
```

This displays Raw image data in hires-interlaced mode.

```
SAVER "OPALVISION" "XXX" "IMAGE" SCREEN_TYPE 8 DURATION 100
```

This displays Rendered image data in lores-overscan for 2 seconds.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	Error string

13.9.24 PCX

Syntax: **SAVER "PCX" *filename type pcx_type***

The *type* argument must be either "RAW" or "IMAGE".

PCX files can come in four flavors: CGA, EGA, VGA, and 24 bit.

If you specify that you wish to save **RAW** data, any *saver_opts* which you might have included will be ignored. The resulting PCX file will either be of the 24 bit-plane color variety (if you have color image data loaded at the time) or will be a 256 shade VGA file (if you have grayscale image data loaded at the time).

If you wish to save **IMAGE** data, a *pcx_type* chosen from one of **CGA**, **EGA**, or **VGA** must be specified, unless the rendered image data available at the time is unambiguously VGA (more than 16 colors).

If you attempt to save an image rendered in a uniquely Amiga mode, an error is returned.

For example, if 256 color rendered image data is available, either of the following two lines are acceptable (to save a VGA file).

```
SAVER "PCX" "test.pcx" "IMAGE"
SAVER "PCX" "test.pcx" "IMAGE" "VGA"
```

And both of the following two lines would produce errors (under the same circumstances).

```
SAVER "PCX" "test.pcx" "IMAGE" "CGA"
SAVER "PCX" "test.pcx" "IMAGE" "EGA"
```

If 16 color rendered image data were available, then the following line would return an error since you can have 16 color VGA and 16 color EGA files.

```
SAVER "PCX" "test.pcx" "IMAGE"
```

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.9.25 PICASSO

Syntax: **SAVER "PICASSO" "XXX" *type saver_opts***

The *type* argument will be ignored in the current version (PICASSO saver v1.9). Only the raw 24 bit or gray image data can be saved with this saver (i.e., use "RAW" for *type*).

The *saver_opts* for the PICASSO saver are:

WIDTH *w*

Set the display x-resolution of the PICASSO II screen (in pixels).

If you specify a width smaller than the picture, then you can scroll through the image with the cursor keys.

GUI Equivalent: Width of the selected screen mode.

Constraints: $1 \leq w$

Required: No

HEIGHT h

Set the display y-resolution of the PICASSO II screen (in pixels).
If you specify a height smaller than the picture, then you can scroll through the image with the cursor keys.

GUI Equivalent: Height of the selected screen mode.

Constraints: $1 \leq h$

Required: No

DEPTH d

Set the depth of the PICASSO II screen to display the picture in.

GUI Equivalent: Depth of the selected screen mode.

Constraints: $d = 15, 16, \text{ or } 24$ (corresponding to 32,000, 64,000, and 16 million colors).

Required: No

If the screen dimensions are smaller than the image, you can scroll through the whole picture with the cursor keys (see the documentation on the PICASSO saver's control panel). If you don't supply any screen dimensions, the PICASSO saver will choose the best matching mode. You must **ALWAYS** supply a filename parameter ("**XXX**") even if it's not used by the PICASSO saver.

Example:

```
SAVER "PICASSO" "XXX" "RAW" WIDTH 320 HEIGHT 240 DEPTH 16
```

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	error message

13.9.26 POSTSCRIPT

The POSTSCRIPT saver lets you export images in PostScript or Encapsulated PostScript format. No ARexx control exists at this time.

13.9.27 PREFPRINTER

Syntax: **SAVER "PREFPRINTER" "XXX" *type saver_opts***

The *type* argument must be "RAW" only.

The *saver_opts* can be the following:

OUTPUT (PREFERENCES | PRIMERA)

Select the print destination, to the Preferences-supported printer or to the FARGO Primera dye-sublimation printer.

THIS MUST BE THE FIRST OPTION SPECIFIED AFTER THE "RAW" argument.

GUI Equivalent: **Print To** selection.

Constraints: None

Required: Yes

ORIENTATION (HORIZONTAL | VERTICAL)

Set the orientation of the image on the printed page.

GUI Equivalent: **Orientation** setting.

Constraints: None

Required: No; default: current orientation

DITHER (FLOYD | ORDERED | DHORIZONTAL | DVERTICAL | FWDBRICK | BCKBRICK | FWDDIAGONAL | BCKDIAGONAL | HALFTONEA | HALFTONEB | ASCII)

Set the dither method to use. This control is ignored when printing to the Primera.

GUI Equivalent: **Dither** setting.

Constraints: None

Required: No; default: previous dither

MASK_SIZE (2 | 4 | 8 | 16)

Set the mask size for the non-Floyd and non-ASCII dither modes. This control is ignored when printing to the Primera.

GUI Equivalent: **Mask Size** setting.

Constraints: Ignored if using FLOYD or ASCII dither.

Required: No; default: previous mask size

(DENSITY *d* | ASCII_OVERSTRIKE *ao*)

Set the print density (DENSITY) or number of times to strike each line (ASCII_OVERSTRIKE). This control is ignored when printing to the Primera.

GUI Equivalent: **Density:** or **ASCII Overstrike:** setting.

Constraints: $0 \leq d \leq 7$
(0 = previous setting);
 $1 \leq ao \leq 5$

Required: No; default: previous setting

PRINT_DIMENSIONS *w h l t*

Set the print dimensions.

GUI Equivalent: **Print Dimensions** settings.

Constraints: None

Required: No; default: previous settings

PAGE_DEFINITIONS *w h l t*

Specify the page definition.

GUI Equivalent: **Page Definitions** settings.
Constraints: None
Required: No; default: previous settings

PAPER_DIMENSIONS *h*

Set the paper height (for printer drivers which support variable-size heights). This control is ignored when printing to the Primera.

GUI Equivalent: **Paper Dimensions** setting.
Constraints: None
Required: No; default: previous setting

(ENGLISH | METRIC)

Set the measuring system to use. This affects the unit of measure used for the **PRINT_DIMENSIONS**, **PAGE_DEFINITIONS**, and **PAPER_DIMENSIONS** arguments. If **ENGLISH**, inches will be the unit of measure; for feet, just divide by 12 and ignore the remainder. Otherwise, centimeters; for decimeters, just divide by 10 and ignore the remainder.

GUI Equivalent: **Paper Dimensions** setting.
Constraints: None
Required: No; default: English

PRIMERA_PAPER (**A_STD** | **A4_STD**)

Set the Primera paper size. This control is ignored when printing to a printer supported by a Preferences printer driver.

GUI Equivalent: **Primera Paper** setting.
Constraints: None
Required: No; default: previous setting

PRIMERA_DITHER (**ON** | **OFF**)

Set the Primera dither mode. If off, output will only be 12-bit (4,096 colors). If on, output will be 24-bit (over 16 million colors). This control is ignored when printing to a printer supported by a Preferences printer driver.

GUI Equivalent: **Primera Dither** setting.
Constraints: None
Required: No; default: use dithering

PRIMERA_SPOOL_DIRECTORY *directory*

Specify the directory where Primera-related temporary files will be stored. A maximum of 5 megabytes may be used. The files PREF-PRINTER places in this directory will be deleted after it finishing printing. This control is ignored when printing to a printer supported by a Preferences printer driver.

The initial setting is **T:**, which is usually assigned to your RAM Disk. If you don't have at least 5 megabytes, you should change this directory to a location on your hard disk which has more space.

GUI Equivalent: **Primera Spool Directory...** selection.

Constraints: None

Required: No; default: **T:** directory

PRIMERA_HEAT *ph*

Set the Primera heat level. This control is ignored when printing to a printer supported by a Preferences printer driver.

GUI Equivalent: **Heat Setting (n)...** value.

Constraints: None

Required: No; default: 49

NOTE The old ARexx syntax is only supported if the first option after the "RAW" argument is **not** OUTPUT. Otherwise, the new syntax (described above) will be assumed.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.9.28 QRT

Syntax: **SAVER "QRT" filename type**

The *type* argument must be "RAW" only.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.9.29 RESOLVER

Syntax: **SAVER "RESOLVER" "XXX" type saver_opts**

The *type* argument must be "IMAGE" only, or "RAW" for 8-bit grayscale images.

The ARexx support for the DMI Resolver saver constitutes a language in-and-of-itself. From ARexx, each call to the DMI Resolver saver defines a stream of commands (defined as *saver_opts*) which are parsed and executed from left to right. These commands are:

BOARD_MODE *num*

This command sets the display size based on the currently defined size numbers. These mode numbers are defined within the DMI configuration software.

If an invalid number is specified, an error will occur and the saver will exit with an error code of 10.

BOARD_NO *num*

If you have more than one DMI Resolver installed, this command selects to which board subsequent commands will be sent. All commands described here operate on the currently selected board.

CENTER

This command sets the display offsets to center the image currently in Amiga memory.

CLEAR

This command clears the currently selected board.

IMAGE

This command causes the image currently in Amiga memory to be downloaded to the currently selected board at the current offsets.

SET_DOX *left_edge*

This command specifies which column (DOX means "Destination Offset X") the image will be written to.

SET_DOY *top_edge*

This command specifies which row (DOY means "Destination Offset Y") the image will be written to.

SET_DWX *width*

This command sets the width of the image area to be written to the specified value. The required argument *width* is in DMI Resolver pixels.

SET_DWY *height*

This command sets the height of the image area to be written to the specified value. The required argument *height* is in DMI Resolver pixels.

SET_SOX *left_edge*

This command specifies the left edge of the image area within the image in memory which will be written to the board (SOX means "Source Offset X").

SET_SOY *top_edge*

This command specifies the top edge of the image area within the image in memory which will be written to the board (SOY means "Source Offset Y").

Since ARexx command lines invoking the DMI Resolver saver can be complex, finding an error on the command line can become difficult. The saver, therefore, returns in **ADPRO_RESULT** the number of the argument in which an error was found. The argument numbering begins with 1, which corresponds to the *type* argument of the ARexx **SAVER** command. If an error is encountered, RC will return non-zero.

For instance, the following returns a 1 in **ADPRO_RESULT**:

```
SAVER "RESOLVER" "XXX" "NEITHER_RAW_NOR_RENDERED" "IMAGE"
```

Examples of normal usage:

```
SAVER "RESOLVER" "XXX" "IMAGE" "CLEAR" "CENTER" "IMAGE"
```

This will clear the board, center, and download the image to the current selected DMI Resolver.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.9.30 RETINA

Syntax: **SAVER "RETINA" "XXX" *type saver_opts***

The *type* argument must be one of "IMAGE", "GRAY", or "RAW".

The *saver_opts* for the RETINA are:

DELAY *d*

Specify a period of seconds for the image to stay on the screen. You must supply the **CLOSE** keyword to remove the image from the screen.

CLOSE

This option will remove a previously displayed image from the screen. This option may also be used to close a screen kept open by another **SAVER** command.

NOTE The Retina will keep an image on the screen even if you quit the saver or ADPro. The screen will be closed if its memory is needed by another image, you click the right mouse button, or if you type **RetinaAvail FLUSH**. See the Retina User's Manual for information on RetinaAvail.

Examples of normal usage:

```
SAVER "RETINA" "XXX" RAW DELAY 5 CLOSE
```

This displays a 24-bit image, waits 5 seconds then closes the screen.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.9.31 SCULPT

Syntax: **SAVER "SCULPT" *filename type***

The *type* argument must be "RAW" only.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.9.32 SEPARATE

Syntax: **SEPARATE *opts***

SAVER "SEPARATE" *rootname plane*

This section describes how the commands relating to color separation are used from ARexx. Note that all image data saved by ADPro's color separation routines are saved in 4 or 8 bit-plane IFF files. If you require saved color separation data in a different format, use ADPro to convert the IFF format file into the format you need.

Before you can do a separation (with the **SAVER** command), you should set up your options with zero or more invocations of the following commands:

UCR (1)

UCR *value* (2)

The first form of the command returns the current UCR (Under Color Removal) setting in **ADPRO_RESULT**.

The second form of the command allows the UCR value to be set to the supplied value which must be in the range of 0 to 100.

If the supplied value is outside the range of 0 to 100, then the **RC** variable will be set to 10 and **ADPRO_RESULT** will be set to the string **Invalid Argument To Function**.

If the supplied value is accepted, **RC** returns as 0 and **ADPRO_RESULT** returns with the *previous* setting.

GCR (1)
GCR *value* (2)

The first form of the command returns the current GCR (Gray Component Replacement) setting in **ADPRO_RESULT**.

The second form of the command allows the GCR value to be set to the supplied value which must be in the range of 0 to 100.

If the supplied value is outside the range of 0 to 100, then the RC variable will be set to 10 and **ADPRO_RESULT** will be set to the string **Invalid Argument To Function**.

If the supplied value is accepted, RC returns as 0 and **ADPRO_RESULT** returns with the *previous* setting.

SEPARATION_MAP (1)
SEPARATION_MAP *value* (2)

The first form of the command returns the current **CMAP** rocker switch setting in **ADPRO_RESULT**. This will be in the form of text reading either **COLOR** or **GRAY**.

The second form allows you to set the value of the **CMAP** rocker switch. The single supplied argument must be either **COLOR** or **GRAY**. If the call was successful, the value returned in **ADPRO_RESULT** is the previous **CMAP** rocker switch setting.

If the call is unsuccessful, the value returned in RC is 10. A value of 0 is returned in RC if the call (either form) completed properly.

MAGENTA_CORRECTION (1)
MAGENTA_CORRECTION *value* (2)

The first form of the command returns the current magenta correction setting in **ADPRO_RESULT**.

The second form of the command allows the magenta correction value to be set to the supplied value which must be in the range of 0 to 100.

If the supplied value is outside the range of 0 to 100, then the RC variable will be set to 10 and **ADPRO_RESULT** will be set to the string **Invalid Argument To Function**.

If the supplied value is accepted, RC returns as 0 and **ADPRO_RESULT** returns with the *previous* setting.

YELLOW_CORRECTION (1)
YELLOW_CORRECTION *value* (2)

The first form of the command returns the current yellow correction setting in **ADPRO_RESULT**.

The second form of the command allows the yellow correction value to be set to the supplied value which must be in the range of 0 to 100.

If the supplied value is outside the range of 0 to 100, then the RC variable will be set to 10 and **ADPRO_RESULT** will be set to the string **Invalid Argument To Function**.

If the supplied value is accepted, **RC** returns as 0 and **ADPRO_RESULT** returns with the *previous* setting.

INK_CORRECTION

(1)

INK_CORRECTION (0 | 1)

(2)

The first form of the command returns the current state of the ink correction setting in **ADPRO_RESULT**.

The second form of the command allows the state of the ink correction setting to be set to the specified state which must either be 0 or 1. A value of 0 disables ink correction. A value of 1 enables ink correction.

If the supplied value is neither 0 nor 1 then the **RC** variable will be set to 10 and **ADPRO_RESULT** will be set to the string **Invalid Argument To Function**.

If the supplied value is accepted, **RC** returns as 0 and **ADPRO_RESULT** returns with the *previous* state of the ink correction setting.

SEPARATION_DEPTH

(1)

SEPARATION_DEPTH (4 | 8)

(2)

The first form of the command returns the current depth setting for color separations. The returned value will be either a 4 or 8 and is returned in **ADPRO_RESULT**.

The second form of the command allows the depth of color separations to be set to the supplied value which must be 4 or 8.

If the supplied value is neither 4 nor 8, then the **RC** variable will be set to 10 and the **ADPRO_RESULT** will be set to the string **Invalid Argument To Function**.

If the supplied value is accepted, **RC** returns as 0 and **ADPRO_RESULT** returns with the *previous* color separation depth setting.

SEPARATION_TYPE

(1)

SEPARATION_TYPE (RGB | CMY | CMYK)

(2)

The first form of the command returns the current separation type. This will be returned in **ADPRO_RESULT** and will be one of the following strings: **RGB**, **CMY**, or **CMYK**.

The second form of the command sets the separation type to the supplied value which must match one of the above mentioned strings.

If the supplied string is not acceptable, then the **RC** variable will be set to 10 and the **ADPRO_RESULT** will be set to the string **Invalid Argument To Function**. If the supplied string is accepted, **RC** is returned as zero, and **ADPRO_RESULT** will contain the *previous* separation type.

USE_SEPARATION_DEFAULTS

This command sets the **UCR**, **GCR**, and ink corrections values to their default settings. This command never returns an error.

SEPARATE *rootname plane*

This command is equivalent to the actual **SAVER "SEPARATE" *rootname plane*** calling sequence. It has been retained for compatibility with current ARexx scripts.

The actual **SAVER** command causes a separation to actually take place. If the separation type is **RGB**, then the *plane* argument may be one or all of *r*, *g*, and *b*. If the separation type is **CMY**, then the *plane* argument may be one or all of *c*, *m*, and *y*. If the separation type is **CMYK**, then the *plane* argument may be one or all of *c*, *m*, *y*, or *k*.

rootname is the name to be used for the separation output *without any file name extensions appended*. ADPro will append the extensions appropriate for the specific type of separation data that it writes.

Do not specify an extension in the supplied file name as ADPro will supply an appropriate extension automatically.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	non-zero value	undefined

13.9.33 TEMP

Syntax: **SAVER "TEMP" "XXX" "RAW"**

The TEMP saver save command places the raw image data into the temporary image buffer.

The TEMP saver does not have any *saver_opts*.

If save was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.9.34 VT_BUFFER

Syntax: **SAVER "VT_BUFFER" "XXX" *saver_opts***

where *saver_opts* can be the following:

BUFFER (1 | 2)

Select the frame buffer to use.

GUI Equivalent: **Buffer** selection.

Constraints: None

Required: No; default: 1 (DV1).

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	error message

13.10 Operators

All of the operators can be controlled through an operator-specific control panel (or screen of panels) or through their ARexx interfaces.

13.10.1 Setting the Operate Format

Syntax: **OFORMAT** (1)
OFORMAT *format* (2)

The first form of the command will fill the **ADPRO_RESULT** field with the name of the currently selected Operator Format, if results are requested.

The second form of the command sets the Operator Format to the specified string.

If module	RC	ADPRO_RESULT
exists	0	the <i>previously</i> selected Operator Format
does not exist	10	"Unknown Module"

13.10.2 Invoking the Current Operator

Syntax: **OPERATE** *operator_args*

This command invokes the currently selected operator (selected with the **OFORMAT** command or set by the last **OPERATOR** invocation) using the given arguments. This command is equivalent to the **Execute Op** button on the control panel.

13.10.3 Invoking an Operator

Syntax: **OPERATOR** *operator_name* [*operator_args*]

The **OPERATOR** command is a general purpose front-end to run-time loadable ADPro operators. Operators are independent programs typically performing an image processing function which modifies the raw and/or rendered image data currently available.

As given above, the generalized syntax of the **OPERATOR** command requires the operator name to be given as the command's first argument. Additional arguments may be required depending upon the particular operator being called.

The following sections detail the operators provided as standard with ADPro.

13.10.4 Launching an Operator

Syntax: `OPERATE_GUI`

This command will launch the currently selected operator as if it were selected manually from ADPro's main window.

Pseudo operator modules **cannot** be executed with this command.

If GUI was	RC	ADPRO_RESULT
was opened	0	undefined
could not be opened	10	"Error"

13.10.5 Antique

Syntax: `OPERATOR "ANTIQUE"`

This operator does not have any operator-specific ARexx arguments.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.10.6 Apply_Map

Syntax: `OPERATOR "APPLY_MAP"`

This operator causes any raw image data to be replaced by the raw image data after filtering through the current color controls.

If raw data is	RC	ADPRO_RESULT
available	0	undefined
not available	10	undefined

13.10.7 Blur

Syntax: `OPERATOR "BLUR" cw th` (1)
`OPERATOR "BLUR" cw th "TEST"` (2)

The first form of the operator will blur any raw data present using the supplied center weight (*cw*) and threshold (*th*).

The second form of the operator will test the supplied values by performing the appropriate calculations but not actually modify the image.

Both forms return the number of pixels which would be (in the second form) or were (as in the first form) affected by the operation in **ADPRO_RESULT**.

The *cw* value may range from 0 to 16; *th* may range from 0 to 255.

If	RC	ADPRO_RESULT
operation was successful	0	undefined
raw data is not available or values are out of range	10	undefined

13.10.8 Broadcast_Limit

Syntax: **OPERATOR "BROADCAST_LIMIT"** (1)
 OPERATOR "BROADCAST_LIMIT" *s* (2)
 OPERATOR "BROADCAST_LIMIT" *s lt* (3)
 OPERATOR "BROADCAST_LIMIT" *s lt other_opts* (4)

The first form will apply the Broadcast_Limit operator to any raw data present using the settings that were previously defined. Optional arguments may be specified to change the settings of the operator.

This operator returns the number of pixels affected by the operation in **ADPRO_RESULT**.

The second form allows you to specify (with *s*) the television standard which will be used in the calculations performed. The *s* should be replaced by either the string "NTSC" or "PAL".

The third form allows you to specify (with *lt*) the limit type. The limit type may be specified as the string "LUMINANCE" or "SATURATION". This has the same effect as toggling the **Method:** cycle gadget on the Broadcast_Limit control panel.

The fourth form allows you to specify additional options which can appear in any order. These include:

"COMPOS_LIM" *c*

This allows you to change the maximum value allowed for the composite signal that would be generated if the image was to be displayed on a NTSC or PAL television. The *c* value has a range from 0.00 to 199.99. If the value given is out of the acceptable range, ADPro will return a result of 10 in the **RC** variable. The **ADPRO_RESULT** variable will contain the argument number (beginning with 1 corresponding to the *s* parameter) which caused the error.

"CHROMA_LIM" *cv*

This allows you to change the maximum value allowed for the chroma in the image. The *cv* value has a range from 0.00 to 99.99. If the given value is out of the acceptable range, ADPro will return a result of 10 in the *RC* variable. The **ADPRO_RESULT** variable will contain the argument number which caused the error (beginning with 1 corresponding to the *s* parameter).

TEST

Specifying this argument allows you to “pencil test” the Broadcast_Limit operation. This allows you to test your settings of the operator, without changing any of the raw data. The number of pixels which would have been affected will be returned in the **ADPRO_RESULT** variable.

DEFAULT_LIM

Specifying this command causes the composite and chroma values to be reset to the default settings, which are 110.0 for composite, and 50.0 for the chroma setting.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	Error argument number

13.10.9 Collapse

Syntax: **OPERATOR "COLLAPSE" arguments**

where *arguments* must be the following (specified in any order):

AMOUNT *a*

Set the degree of collapse.

GUI Equivalent: **Amount (Deg)** value.

Constraints: $0 \leq a \leq 100.0$

Required: Yes

BLUR_RADIUS *b*

Set the percent of the radius (starting at the outer part of the circle) that will be blurred.

GUI Equivalent: **BRadius(0-99)** value.

Constraints: $0 \leq b \leq 99$

Required: Yes

CENTER *xpos ypos*

Set the location of the collapse circle's center.

GUI Equivalent: **Center X (Px)** and **Center Y (Px)** values.

Constraints: $-30000 \leq xpos \leq 30000$; $-30000 \leq ypos \leq 30000$

Required: Yes

RADIUS *r*

Set the radius of the collapse circle.

GUI Equivalent: **Radius (Px)** value.

Constraints: $5 \leq r \leq 15000$

Required: Yes

(QUALITY_FAST | QUALITY_HIGH)

Set the precision of the collapse operation.

GUI Equivalent: **Quality** rocker switch setting.

Constraints: None

Required: Yes

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	Error message

13.10.10 Color_To_Gray

Syntax: OPERATOR "COLOR_TO_GRAY" (1)

OPERATOR "COLOR_TO_GRAY" *rw gw bw* (2)

This operator converts raw color image data into raw grayscale image data.

This operator requires raw color data.

The first form of the command converts color information into grayscale using the weighting factors used by the NTSC standard.

The second form allows you to specify the weighting to be used for the red, green, and blue components of the image. Weights are scaled to a range of 0 for a zero weight to 10000 for a 100 percent weight. Weights may be negative or exceed 100 percent.

If raw color data is	RC	ADPRO_RESULT
available	0	undefined
not available	10	"Error"

13.10.11 Colorize

Syntax: OPERATOR "COLORIZE" *t pf pt* (1)

OPERATOR "COLORIZE" *t pf pt f* (2)

OPERATOR "COLORIZE" *t pf pt f r g b* (3)

OPERATOR "COLORIZE" *t pf pt f r g b htp* (4)

OPERATOR "COLORIZE" *t pf pt f r g b htp hf ht* (5)

This operator requires the presence of raw grayscale image data which it converts into raw color data where some or all of the gray image has actually been colored in.

The first form allows you to specify a range of grayscales which will be affected by the colorizing operation. Note that this range always spreads numerically upwards from the *pf* value to the *pt* value (i.e., from and to) and that, if necessary, it will wrap back to 0 if it reaches 255.

The second form allows you to specify (with *f*) what is to be done with those shades of gray which lie outside the range you specify with *pf* and *pt*. If the *f* value is set to "-1" (note that the quotation marks are required when specifying a negative number), then the gray shades are passed through without change. If *f* is set to a value between 0 and 255 (inclusive), pixels outside the range specified with *pf* and *pt* are set to the gray shade specified by *f*.

The third form allows you to set the base R, G and B color used by the HSV and RANDOM colorizations.

The fourth form allows you to select which type of HSV colorization you wish to perform. The argument *htp* is valid only when *t* is set to HSV. This argument may be one of VALUE, HUE, or SATURATION.

The fifth form allows you to specify the range of hues, values, or saturations which will be used during an HSV colorization. The *hf* and *ht* arguments are valid only when *t* is set to HSV. The *hf* and *ht* range depends upon the type of HSV colorization being specified (in the *htp* variable).

hf

If *htp* is "VALUE", this may be between 0 and 1.

If *htp* is "SATURATION", this may be between 0 and 1.

If *htp* is "HUE", this may be between 0 and 10000.

ht

If *htp* is "VALUE", this may be between 0 and 1.

If *htp* is "SATURATION", this may be between 0 and 1.

If *htp* is "HUE", this may be between 0 and 10000.

This operator will set RC to non-zero if an error has occurred or will return 0 in RC if the operation was successful.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.10.12 Convolve

Syntax: OPERATOR "CONVOLVE" *matrix_file mix thresh*

This operator causes the convolution matrix contained in the file named *matrix_file* to be passed over the current image data using the value of *mix* as the mixing setting and the value of *thresh* as the threshold setting.

NOTE The string you pick for *matrix_file* must be a full path name to the matrix file you wish to use. For example, you might specify "ADPRO:Convolutions/MatrixName".

The valid range of *mix* settings is 1 to 100. The valid range of *thresh* is from 0 to 255.

If the operation is aborted by the user, RC will return as 10 and ADPRO_RESULT will contain the string **Aborted**. If the operation succeeded, RC will return as a value of 0 and ADPRO_RESULT will contain the name of the operator which was used previous to this one.

If operation was	RC	ADPRO_RESULT
successful	0	<i>previousoperator</i>
aborted	10	Aborted

13.10.13 Crop_Image

Syntax: OPERATOR "CROP_IMAGE" *width height* (1)
 OPERATOR "CROP_IMAGE" *width height left top* (2)

This operator can be used to select and preserve a rectangular region within an image and discard all data outside the selected region.

The first form of the command preserves the specified *width* and *height* starting at the upper left-hand corner of the original image.

The second form allows you to specify an offset (*left* and *top*) relative to the top left-hand corner of the original image.

Note that the offsets (if specified) cannot be negative nor can the offsets plus the width or height exceed the width or height of the original image. If the supplied parameters are not accepted, then the RC value will be set to 10. If the operation is successful, RC is returned with a value of 0.

From ARexx, you should use this operator rather than Crop_Visual.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.10.14 Crop_Visual

Syntax: **OPERATOR "CROP_VISUAL"** *width height* (1)
 OPERATOR "CROP_VISUAL" *width height left top* (2)

This operator can be used to select and preserve a rectangular region within an image and discard all data outside the selected region.

The first form lets you specify the width and height of the cropped area using an offset of 0,0 (the upper left corner of the original image).

The second form lets you specify the width, height, and left and top offsets.

Note that if you specify offsets (second form), they cannot be negative nor can the offsets plus the width or height exceed the width or height of the original image.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.10.15 DCTV

Syntax: **OPERATOR "DCTV"**

This operator converts the currently loaded raw image data into rendered data usable by the DCTV display device. You can use the IFF saver to save the converted data to disk or you can use the **ADPRO_DISPLAY** command to view the DCTV data from within ARexx.

The DCTV operator takes its display size and type (i.e., interlace on or not, PAL or NTSC, etc.) from ADPro's current screen control settings.

If ADPro is currently set to render 8 colors, then a three bit-plane DCTV image will be created. If ADPro is set to render any other type of image (not 8 colors), then a four bit-plane DCTV image will be created.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.10.16 Define_Pxl_Aspect

Syntax: **OPERATOR "DEFINE_PXL_ASPECT"** *arguments*

where *arguments* can be zero or more of the following (listed in any order):

XASPECT *xasp*

Set the horizontal pixel aspect.

GUI Equivalent: **New (X) Aspect:** value.

Constraints: $1 \leq xasp \leq 240$

Required: No; default: *xaspect* = current setting

YASPECT *yasp*

Set the vertical pixel aspect.

GUI Equivalent: **New (Y) Aspect:** value.

Constraints: $1 \leq yasp \leq 240$

Required: No; default: *yasp* = current setting

XRES *xres*

Set the horizontal resolution (DPI).

GUI Equivalent: **X DPI:** value.

Constraints: $1 \leq xres \leq 9999$

Required: No; default: *xres* = current setting

YRES *yres*

Set the vertical resolution (DPI).

GUI Equivalent: **Y DPI:** value.

Constraints: $1 \leq yres \leq 9999$

Required: No; default: *yres* = current setting

This operator allows you to define the current image's pixel aspect and resolution information. If no arguments are given, then the current aspect and resolution information will be returned in a string (**ADPRO_RESULT**) of the following form:

xasp yasp xres yres width height

where *xasp* and *yasp* are the *x* and *y* aspect of the pixels in the currently defined image. The *xres* and *yres* values are the currently defined *x* and *y* resolutions. The *width* and *height* values are the width and height of the image in pixels. You can determine the size (in inches) that this image is asking to be by dividing the width or height values by the *xres* or *yres* fields, respectively.

If	RC	ADPRO_RESULT
no values are given	0	the <i>current</i> pixel aspect and resolution information
values are correct	0	the <i>previous</i> pixel aspect and resolution information
any of the values are out of range or don't make sense	non-zero	undefined

13.10.17 DeInterlace

Syntax: OPERATOR "DEINTERLACE"

This operator splits the currently defined image into two images, stacked one above the other. In the top half, the even numbered scanlines of the former image can be found. In the bottom half, what were formerly the odd numbered scanlines can be found.

This operator is particularly useful for those video digitizing moving scenes.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.10.18 Displace_Pixel

Syntax: OPERATOR "DISPLACE_PIXEL" (1)

OPERATOR "DISPLACE_PIXEL" *rad* (2)

OPERATOR "DISPLACE_PIXEL" *rad prob* (3)

OPERATOR "DISPLACE_PIXEL" *rad prob seed* (4)

The first form performs the operation using the previously-defined settings.

The second form takes a value for the displacement circle's radius (*rad*), but uses the other attributes' current values. The *rad* value can be between 1 and half of the image's width.

The third form takes the *rad* and probability (*prob*) values, but uses the other attributes' current values. The *prob* value can be between 1 and 100.

The fourth form requires that you specify all of the attributes' values, *rad*, *prob*, and the randomizer's *seed* value. This can range between -99999999 and 99999999.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	the previous operator

13.10.19 Dynamic_Range

Syntax: OPERATOR "DYNAMIC_RANGE" (1)

OPERATOR "DYNAMIC_RANGE" *newmin newmax* (2)

The first form of this operator examines any currently loaded raw image data and returns the minimum and maximum values it finds in ADPRO_RESULT. It does not change the data in any way.

The second form of the command remaps any currently loaded raw image data to lie between the supplied new minimum and maximum. These values must be between 0 and 255 inclusively. The old minimum and maximum is returned in **ADPRO_RESULT**.

In both forms, when a maximum and minimum are returned in **ADPRO_RESULT**, they are returned in the following format:

min max

If the operator executed properly, the **RC** variable is set to 0. If an error is found, the **RC** variable is set to 10 and the **ADPRO_RESULT** variable is not set.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.10.20 Gray_To_Color

Syntax: **OPERATOR "GRAY_TO_COLOR"**

This operator converts grayscale raw image data into raw color image data. The resulting raw image has identical red, green and blue components.

If	RC	ADPRO_RESULT
successful	0	undefined
raw grayscale image data is not present when this operator is executed or there is not enough memory available to perform the conversion	10	undefined

13.10.21 Halve

Syntax: **OPERATOR "HALVE"**

This operator scales both the width and height of the currently defined image to one half their original sizes. It does so much more quickly than ADPro's general purpose scaling function but with the same precision.

This operator requires raw data.

If operation	RC	ADPRO_RESULT
was successful	0	undefined
wasn't successful (no raw data available)	10	"Error"

13.10.22 Hist_Equalization

Syntax: OPERATOR "HIST_EQUALIZATION"

This operator requires raw data.

If	RC	ADPRO_RESULT
successful	0	undefined
no raw data is available	10	"Error"

13.10.23 Horizontal_Flip

Syntax: OPERATOR "HORIZONTAL_FLIP"

This operator flips the available raw data horizontally about its center.

This operator requires raw data.

If	RC	ADPRO_RESULT
successful	0	undefined
no raw data is available	10	"Error"

13.10.24 Intensity_Range

Syntax: OPERATOR "INTENSITY_RANGE"

If operation was	RC	ADPRO_RESULT
successful	0	the <i>previous</i> operator
not successful	10	Error

13.10.25 Interlace

Syntax: OPERATOR "INTERLACE"

This operator performs an interlacing function by assuming that the currently loaded image actually contains two separate images (of the same height) stacked one upon the other.

It produces raw image data which contains scanlines chosen alternately from the top and bottom halves of the previously loaded raw image data.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.10.26 KillTemp

Syntax: **OPERATOR "KILLTEMP"**

This operator causes any image data currently in the temporary buffer to be erased.

If raw data is	RC	ADPRO_RESULT
available	0	undefined
not available	10	Error message

13.10.27 Line_Art

Syntax: **OPERATOR "LINE_ART"**

This operator causes a proprietary edge detection algorithm to be passed over the current raw grayscale image data. The resulting image can sometimes be very good quality line art.

This operator requires raw grayscale data. If no raw grayscale data is available, **RC** will be set to 10 and **ADPRO_RESULT** will be set to **Error**. This operator sets **RC** to a value of 0 to signify that no error occurred.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	Error

13.10.28 Median_Filter

Syntax: **OPERATOR "MEDIAN_FILTER" *th*** (1)

OPERATOR "MEDIAN_FILTER" *th* "TEST" (2)

The first form will apply the Median_Filter operator to any raw data present using the supplied threshold (*th*).

The second form of the operator will test the supplied threshold by performing the appropriate calculations but not actually modify the image.

Both forms return the number of pixels which would be (in the second form) or were (as in the first form) affected by the operation in **ADPRO_RESULT**.

th may range from 0 to 255. If *th* lies outside the specified range, or there is no raw data available, a value of 10 will be placed in **RC**. Otherwise, **RC** returns set to 0.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.10.29 Mosaic

Syntax: OPERATOR "MOSAIC" *arguments*

where *arguments* can be one or more of the following (specified in any order):

XSIZE *width*

Sets the width of each "tile".

GUI Equivalent: **Width:** value.

Constraints: $1 \leq width$

Required: Yes

YSIZE *height*

Sets the height of each "tile".

GUI Equivalent: **Height:** value.

Constraints: $1 \leq height$

Required: Yes

XOFFSET *xoff*

Sets the left position of the upper-left tile with respect to the left border of the image.

GUI Equivalent: **X Offset:** value.

Constraints: $0 \leq xoff$

Required: No

YOFFSET *yoff*

Sets the top position of the upper-left tile with respect to the top border of the image.

GUI Equivalent: **Y Offset:** value.

Constraints: $0 \leq yoff$

Required: No

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	Error message

13.10.30 Negative

Syntax: OPERATOR "NEGATIVE"

This operator inverts whatever raw image data is available. If raw grayscale data is available, this operator produces a "black and white negative." If color raw data is available, then this operator produces a "color negative".

This operator requires raw data.

If operation	RC	ADPRO_RESULT
was successful	0	undefined
wasn't successful (no raw data available)	10	"Error"

13.10.31 OpalPaint

Syntax: **OPERATOR "OPALPAINT"** *OpalPaint_ARexx_commands*

This operator can be sent OpalPaint-specific ARexx commands.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.10.32 Pattern

Syntax: **OPERATOR "PATTERN"** *arguments*

where *arguments* can be one or more of the following (specified in any order):

FILE *filename*

Specify the pattern to use.

GUI Equivalent: **Pattern...** selection.

Constraints: None

Required: Yes

WIDTH *w*

Set the width of the pattern. This **must** be the same as the pattern's actual width in pixels.

GUI Equivalent: **Pattern Width:** value.

Constraints: $1 \leq w$

Required: Yes

HEIGHT *h*

Set the height of the pattern. This **must** be the same as the pattern's actual height in pixels.

GUI Equivalent: **Pattern Height:** value.

Constraints: $1 \leq h$

Required: Yes

METHOD *m*

Set the pattern method to use. 0 = **B→Spec W→Key**, 1 = **B→Orig W→Key**, 2 = **B→Key W→Orig**, 3 = **B→Key W→Spec**, 4 = **B→Spec W→Orig**, 5 = **B→Orig W→Spec**

GUI Equivalent: **Method:** setting.

Constraints: $0 \leq m \leq 5$

Required: No; default: current setting

COLOR *r g b*

Set the key color, if the pattern method being used requires a value.

GUI Equivalent: **Red:**, **Green:**, and **Blue:** values.

Constraints: $0 \leq r \leq 255$;
 $0 \leq g \leq 255$;
 $0 \leq b \leq 255$

Required: No; default: current setting

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	Error message

13.10.33 Polar_Mosaic

Syntax: **OPERATOR** "POLAR_MOSAIC" *arguments*

where *arguments* can be one or more of the following (specified in any order):

TRACKS *t*

Set the number of circular tracks to create.

GUI Equivalent: **Tracks** value.

Constraints: $1 \leq t \leq 10000$

Required: Yes

BLUR_RADIUS *b*

Set the percent of the radius (starting at the outer part of the circle) that will be blurred.

GUI Equivalent: **BRadius(0-99)** value.

Constraints: $0 \leq b \leq 99$

Required: Yes

CENTER *xpos ypos*

Set the location of the mosaic circle's center.

GUI Equivalent: **Center X (Px)** and **Center Y (Px)** values.

Constraints: $-30000 \leq xpos \leq 30000$; $-30000 \leq ypos \leq 30000$

Required: Yes

RADIUS *r*

Set the radius of the mosaic circle.

GUI Equivalent: **Radius (Px)** value.

Constraints: $5 \leq r \leq 15000$

Required: Yes

SLICES *s*

Set the number of "pie-style" divisions in the circle.

GUI Equivalent: **Slices** value.

Constraints: $1 \leq s \leq 10000$

Required: Yes

(QUALITY_FAST | QUALITY_HIGH)

Set the precision of the mosaic operation.

GUI Equivalent: **Quality** rocker switch setting.

Constraints: None

Required: Yes

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	Error message

13.10.34 Rectangle

Syntax: OPERATOR "RECTANGLE" *le te w h* (1)

OPERATOR "RECTANGLE" *le te w h thk* (2)

OPERATOR "RECTANGLE" *le te w h thk r g b* (3)

OPERATOR "RECTANGLE" *le te w h thk r g b mix* (4)

This operator draws a rectangle into the currently defined raw image. Required arguments include the left edge, top edge, width, and height of the rectangle. The default values for the other arguments (if not specified) are: thickness of 1, color of black, and a mix of 100 percent.

Specifying a thickness of "-1" indicates a filled rectangle. Note that you must include the -1 in quotation marks to prevent ARexx from thinking you wanted to subtract one from the previous argument.

Rectangle color must always be specified by an R, G, B triple. For drawing into a grayscale image, the red value is used.

The mix level must range between 1 and 100.

If any of the supplied arguments are invalid, or no raw image data is available to draw into, the RC variable is set to 10. Otherwise, it is set to 0 and the rectangle is drawn.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.10.35 Rectangle_Visual

Syntax: OPERATOR "RECTANGLE_VISUAL" *arguments*

This operator draws a rectangle into the currently defined raw image. However, we do not recommend the use of this operator from ARexx due to its size. This operator is best used from the user interface because of its visual nature. To draw rectangles from ARexx, use Rectangle instead.

If you insist on using this operator from ARexx, its programming information and behavior is exactly the same as Rectangle.

13.10.36 Rem_Isolated_Pxls

Syntax: OPERATOR "REM_ISOLATED_PXLS"

This operator examines every pixel in a set of rendered data comparing each pixel to its neighbors. If all of a given pixel's neighbors are the same color and that color is different from the given pixel, the pixel is changed to match the color of its neighbors.

This operator requires rendered data. If no rendered data is available, RC will be set to 10 and ADPRO_RESULT is undefined. This operator sets RC to a value of 0 to signify that no error occurred.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.10.37 Rendered_To_Raw

Syntax: OPERATOR "RENDERED_TO_RAW"

This operator converts the rendered image data currently loaded in ADPro's memory back into raw data. The type of the raw data produced (color or grayscale) depends upon which type of rendered image was converted.

If	RC	ADPRO_RESULT
successful	0	undefined
not successful (not enough memory for the raw data)	10	undefined

13.10.38 Roll

Syntax: OPERATOR "ROLL" *direction pixels* (1)

OPERATOR "ROLL" *direction pixels* ("WRAP" | "NO_WRAP") (2)

The first form will apply the Roll operator to any raw data present. The minimum requirements are to specify a roll *direction* and the number of *pixels* to roll the picture.

Substitute one of the following strings for *direction*: "DOWN", "LEFT", "UP", "RIGHT", "UPLEFT", "UPRIGHT", "DOWNRIGHT", or "DOWNLEFT".

The second form will allow you to specify whether or not the image will wrap as it rolls. If "WRAP" is specified, any portion of the image moved off the screen on one side, will appear in the vacated space on the other side. If "NO_WRAP" is specified, the vacated space is filled with black.

If operation	RC	ADPRO_RESULT
was successful	0	undefined
wasn't successful (no raw data available)	10	"Error"

13.10.39 Rotate

Syntax: **OPERATOR "ROTATE" arguments**

where *arguments* must be the following (specified in any order):

CENTER *xpos ypos*

Set the location of the rotate circle's center.

GUI Equivalent: **Center X (Px)** and **Center Y (Px)** values.

Constraints: $-30000 \leq xpos \leq 30000$; $-30000 \leq ypos \leq 30000$

Required: Yes

RADIUS *r*

Set the radius of the rotate circle.

GUI Equivalent: **Radius (Px)** value.

Constraints: $5 \leq r \leq 15000$

Required: Yes

AMOUNT *t*

Set the amount of rotation (in degrees).

GUI Equivalent: **Amount (Deg)** value.

Constraints: $-30000.0 \leq t \leq 30000.0$

Required: Yes

BLUR_RADIUS *b*

Set the percent of the radius (starting at the outer part of the circle) that will be blurred.

GUI Equivalent: **BRadius(0-99)** value.

Constraints: $0 \leq b \leq 99$

Required: Yes

(QUALITY_FAST | QUALITY_HIGH)

Set the precision of the rotate operation.

GUI Equivalent: **Quality** rocker switch setting.

Constraints: None

Required: Yes

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	Error message

13.10.40 Saturation

Syntax: **OPERATOR "SATURATION"** *type amt*

The saturation operator only has one form for use in ARexx programs. *type* may be one of "HSV", "HLS", or "YUV". The *amt* is the amount of adjustment you would like to apply to the image. *amt* has a range of -100 to 9999; any other values will result in an error.

Remember that any negative numbers used as arguments in ARexx must be surrounded by quotes.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.10.41 Scale

Syntax: **ABS_SCALE** *abs_width abs_height* (1)

PCT_SCALE *pct_width pct_height* (2)

The first command scales the raw bit-map presently loaded into ADPro to the supplied dimensions. It is equivalent to the **Variable Enlargement / Reduction** values in the control panel.

The second command scales the raw bit-map presently loaded into ADPro by the supplied percentages. It is equivalent to the **Percent Enlargement / Reduction** values in the control panel.

You can specify any amount of enlargement or reduction so long as the resulting image still fits in memory.

Especially important when doing expansions, ADPro confirms it has enough memory available to perform the desired operation before actually starting to scale. If enough memory is not available, an error will be returned immediately.

While the ADPro user interface will not allow an enlargement and a reduction to be specified at the same time, the ARexx interface does. If the values you specify would cause one axis to enlarge while the other would reduce, both operations will be executed before returning to your ARexx program.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.10.42 Sim Print

Syntax: **OPERATOR "SIM_PRINT"** (1)
 OPERATOR "SIM_PRINT" *method* (2)
 OPERATOR "SIM_PRINT" *method size* (3)

This operator simulates the printing process, using the current raw image in ADPro's memory as its printing "surface". Specifically, an image that would have normally been dithered (to produce higher color fidelity) for paper output would itself be modified with the dither. Unlike other operators, the processing will not affect the original raw image data—the rendered image will be the simulated print.

There can be 0, 1, or 2 arguments given when calling this operator. The first form (0 arguments) will use the currently defined dithering method and size.

The second form (1 argument, *method* only) will use the given dithering method with the currently defined size.

The third form (2 arguments, *method* followed by *size*) will use both the given dithering method and size.

The *method* argument is a string describing the dithering method to use on the image. It can be one of "Halftone", "Ordered", "Vertical Line", "Horizontal Line", "Fwd Diagonal Line", "Bwd Diagonal Line", "Fwd Brick", or "Bwd Brick". These choices have the same spelling as those found using the **Dither:** cycle gadget in this operator's control panel.

NOTE Strings that include space characters must be surrounded by single quotes for ARexx to treat the sequence of words as one whole string. For example: "Vertical Line" must be called as "'Vertical Line'" (note the single quote characters surrounding the string). Also, the string's case (upper and lower) does not have to be the same as shown above for it to be recognized as a valid choice.

Additionally, if you are familiar with the PREFPRINTER saver's *dither_mode* choices, you can also use one of the following similar strings: "HALFTONEA",

"ORDERED", "VERTICAL", "HORIZONTAL", "FWDDIAGONAL", "BCKDIAGONAL", "FWDBRICK", or "BCKBRICK". Note that the "VERTICAL" and "HORIZONTAL" strings are spelled differently than their PREFPRINTER saver counterparts, "DVERTICAL" and "DHORIZONTAL".

The *size* argument is a number describing the size of the dither mask to use. It can be one of 2, 4, 8, or 16.

For a complete description of these dithering methods and mask sizes, see the PREFPRINTER saver section.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.10.43 Text_Visual

Syntax: **OPERATOR** "TEXT_VISUAL" *arguments*

This operator provides a method of adding anti-aliased text to your imagery using several drawing methods. Unlike some other operators, the arguments to this operator are keyword based. The allowable keywords and their parameters include:

CENTER_XOFFSET

This command, which takes no arguments, causes the text to be centered horizontally within the currently defined image without changing the text's vertical position. Note that this command must actually render a temporary version of the text to determine its horizontal size. Therefore, this command should be executed after all font specification commands have been performed.

This command will cause an error if any of the specified font cannot be opened.

CENTER_YOFFSET

This command, which takes no arguments, causes the text to be centered vertically within the currently defined image without changing the text's horizontal position.

DRAW

This command, which takes no arguments, causes the text to be rendered into the currently defined image using the settings which are in-place at this moment.

During the execution of this command, all requested font related information such as font name and size are checked for validity. If the desired font cannot be located, this command will cause an error.

EMBOSS_DIRECTION *style*

This command defines the *style* of embossing which will take place at the next **DRAW** command. The *style* parameter must be one of **OFF**, **IN**, **OUT**, **N**, **NE**, **NW**, **S**, **SE**, **SW**, **E**, or **W**. A *style* of **OFF** disables embossing.

If the *style* parameter is missing, or is not one of the above, an error will be returned.

FONT_DIR *directory_name*

This command tells the Text_Visual operator to look for your fonts in the supplied directory. Normally, fonts are sought in **FONTS:** only.

All validity checking concerning the choice of font name and size are deferred until the next **DRAW** command. This command will generate an error if *directory_name* is not supplied.

Note that the Text_Visual operator maintains two separate sets of font settings and attributes (one for bit-mapped, and one for scalable fonts) including this attribute.

FONT_NAME *fontname*

This command causes the named font to be selected. Note that all validity checking concerning the choice of font name and size are deferred until the next **DRAW** command. Therefore, this command will not cause an immediate error if the requested font does not exist. However, if the font name parameter is missing entirely, an error will be generated immediately.

Note that the Text_Visual operator maintains two separate sets of font settings and attributes (one for bit-mapped, and one for scalable fonts) including this attribute.

FONT_TYPE *type*

This command specifies which *type* of font should be looked for (under the font name given elsewhere in the command stream). *type* may be either **SCALED** or **BITMAPPED**. If the *type* parameter is missing or is not one of the above, an error is generated.

LOAD_SETUP *setup_file*

This command causes the file specified by *setup_file* to be loaded to define various settings. If the *setup_file* parameter is missing, or if an error occurs in loading the settings from the file specified by *setup_file*, an error will be generated.

RENDER_TYPE *rtype*

This command defines which drawing mode will be used to render the text into the currently defined image. The valid choices for *rtype* are: **MIX**, **DARKEN**, **LIGHTEN**, **TRANSPARENT**, and **OUTLINE**.

If the *rtype* parameter is missing or is not one of the above, an error will be generated.

SET_BLUR *bfactor*

This command sets the amount of blurring which will take place. A *bfactor* of "-1" turns blurring off. Other allowable values range from 0 to 16 where 0 produces the greatest blurring. An error is generated if the supplied *bfactor* is out of range.

SET_COLORS *r g b*

This command sets the color used in the rendering of the text. The three arguments *r*, *g*, and *b* must all be present and must range from 0 to 255, otherwise an error is generated.

SET_EMOSS *level*

This command sets the *level* of embossing which must range from 1 to 100. If you wish not to emboss at all, use the **EMBOSS_DIRECTION** command to turn embossing completely **OFF**. An error is generated if the required parameter *level* is missing or is out of range.

SET_FONT_SIZE *size*

This command specifies the *size* of the font to be used for drawing. Note that no validity checking is done at the time this command executes to ensure that the requested size is available. An error will be generated if the *size* parameter, which must range from 1 to 127, is missing.

Note that the Text_Visual operator maintains two separate sets of font settings and attributes (one for bit-mapped, and one for scalable fonts) including this attribute.

SET_FONT_XASPECT *xasp*

This command sets the X aspect of the font to the given value.

SET_FONT_YASPECT *xasp*

This command sets the Y aspect of the font to the given value.

SET_RENDER *level*

This command sets the *level* to be used in drawing in the style you specified with the **RENDER_TYPE** command. *level* must be present and must range from 1 to 100, otherwise an error is generated.

SET_SATURATION *level*

This command sets the *level* of the saturation based drawing effect. *level* must be present and can range from 0 to 100. Specifying a value of 0 will disable saturation based drawing. If *level* is missing, or is out of range, an error will be generated.

SET_TEXT_STYLE *style*

This command sets the *style* of the typeface. The *style* parameter must be present and is a number between 0 and 15. Values used in constructing this number are given in Table 13.3. A *style* of 0 specifies the plain style.

For example, a *style* of 6 specifies bold and italic together.

SET_TINT *level*

This command sets the *level* of the tint based drawing effect. *level* must be present and can range from 0 to 100. Specifying a value of 0 will disable tint based drawing. If *level* is missing, or is out of range, an error will be generated.

Style	Value
Underlined	1
Bold	2
Italic	4
Extended	8

Table 13.3: Values used in constructing a `TEXT_STYLE`.

`SET_TRACKING` *tracking*

This command sets the *tracking*, or space between characters. The *tracking* parameter must be present and must fall between -127 and 1000. If the value specified by *tracking* is out of range, or is missing, an error will be generated.

`SET_XOFFSET` *column*

This command determines the horizontal position of your text based upon how you have defined your `TEXT_HANDLE`. See the description of `TEXT_HANDLE` below for more information. The *column* parameter must be present.

`SET_YOFFSET` *row*

This command defines the vertical position of your text. The specified *row* will correspond to the top-most row of the drawn text. The *row* parameter must be present.

`STRING` *str*

This command sets the string to be drawn. The *str* parameter must be present and must be enclosed in quotes.

`TEXT_HANDLE` *pos*

This command defines how the value specified with the `SET_XOFFSET` command relates to your string. *pos* must be present and must be one of `LEFT`, `CENTER` or `RIGHT`. If `TEXT_HANDLE` has been set to `LEFT`, then your string will begin at the value specified with `SET_XOFFSET`. If set to `CENTER`, then your string will be centered about the position specified by `SET_XOFFSET`. If your handle is set to `RIGHT`, then your text will end at the column specified by `SET_XOFFSET`.

If you don't specify this command as part of your command stream, your handle position will be taken from its last defined value.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.10.44 Tile

Syntax: **OPERATOR "TILE"** *le te w h* (1)

OPERATOR "TILE" *le te w h type amt* (2)

This operator tiles the currently defined bit-map with the specified rectangle. In both forms, the required arguments include *le* (the left edge of the rectangular area), *te* (the top edge of the rectangular area), *w* (the width), and *h* (the height).

The first form produces a tiling with no skew.

The second form allows you to specify which *type* of skew (either the string **HORIZONTAL** or the string **VERTICAL**) and the amount (*amt*) of skew.

If the tiling operation took place, a value of 0 is returned in **RC**. If an error was detected (for example, if the values supplied for *le*, *te*, *w*, and *h* define a rectangle which extends beyond the border of the currently defined image), **RC** is set to 10 and **ADPRO_RESULT** is undefined.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

13.10.45 Tile_Visual

Syntax: **OPERATOR "TILE_VISUAL"** *arguments*

This operator tiles the currently defined bit-map with the specified rectangle. However, we do not recommend the use of this operator from **ARexx** due to its size. This operator is best used from the user interface because of its visual nature. To tile from **ARexx**, use **Tile** instead.

If you insist on using this operator from **ARexx**, its programming information and behavior is exactly the same as **Tile**. However, we again suggest that you do not use this operator from **ARexx** since doing so will waste time and gain you nothing over using the **Tile** operator.

13.10.46 TPort_Controller

Syntax: **OPERATOR "TPORT_CONTROLLER"** (1)

OPERATOR "TPORT_CONTROLLER" *nframes* (2)

This operator attempts to locate a currently running instance of the MicroIllusions Transport Controller. If found, **ADPro** will request that the Transport Controller record the currently rendered image.

The first form of the command causes the Transport Controller to record its default number of frames.

The second form of the command causes the Transport Controller to record the supplied number of frames. When ADPro returns to ARexx, the specified number of frames have been recorded.

This operator requires Amiga displayable rendered data. If no Amiga displayable rendered data is available, RC will be set to 10 and ADPRO_RESULT will be set to **Error**. This operator sets RC to a value of 0 to signify that no error occurred.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	Error

13.10.47 Twirl

Syntax: **OPERATOR "TWIRL" arguments**

where *arguments* must be the following (specified in any order):

CENTER *xpos ypos*

Set the location of the twirl circle's center.

GUI Equivalent: **Center X (Px)** and **Center Y (Px)** values.

Constraints: $-30000 \leq xpos \leq 30000$; $-30000 \leq ypos \leq 30000$

Required: Yes

RADIUS *r*

Set the radius of the twirl circle.

GUI Equivalent: **Radius (Px)** value.

Constraints: $5 \leq r \leq 15000$

Required: Yes

AMOUNT *t*

Set the amount of twirl (in degrees).

GUI Equivalent: **Amount (Deg)** value.

Constraints: $-30000 \leq t \leq 30000$

Required: Yes

BLUR_RADIUS *b*

Set the percent of the radius (starting at the outer part of the circle) that will be blurred.

GUI Equivalent: **BRadius(0-99)** value.

Constraints: $0 \leq b \leq 99$

Required: Yes

(QUALITY_FAST | QUALITY_HIGH)

Set the precision of the twirl operation.

GUI Equivalent: **Quality** rocker switch setting.

Constraints: None

Required: Yes

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	Error message

13.10.48 Vertical_Flip

Syntax: OPERATOR "VERTICAL_FLIP"

This operator flips the available raw data vertically about its center.

This operator requires raw data.

If	RC	ADPRO_RESULT
successful	0	undefined
no raw data is available	10	"Error"

13.11 Displays

Display modules cannot be directly selected from the main window. On startup, ADPro will search the **Displays2** directory for modules. If your graphics hardware can support them, the display modes will be available in the Set Render Screen requester.

This section includes information on displaying and working with image data, as well as technical information on these display modules.

13.11.1 Number of Colors in Rendered Image

Syntax: **RENDER_TYPE** (1)
RENDER_TYPE *rvalue* (2)

The first form of the command returns the current rendering type. This result will be returned in **ADPRO_RESULT** and may be one of the following: 2, 4, 8, 16, 32, 64, 128, 256, EHB, HAM, HAM8, or CUST.

The second form of the command allows the rendering type to be set to the supplied string which must match one of the above mentioned values.

If string was	RC	ADPRO_RESULT
accepted	0	the <i>previous</i> rendering type
not accepted	10	"Invalid Argument To Function"

For more information about rendering type, please see Section 5.6.2.

13.11.2 Rendering an Image

Syntax: **EXECUTE**

This command is the analog of the **Execute** button on the ADPro main window. It causes rendered data to be created from raw data.

If operation was	RC	ADPRO_RESULT
successful	0	undefined
not successful	10	undefined

For more information about the **Execute** button, please refer to Section 5.6.4.

13.11.3 Displaying an Image

Syntax: **ADPRO_DISPLAY**

This command is equivalent to the **Redisplay** button on the ADPro main window. Executing this command will cause any Amiga displayable rendered image to pop to front. Images rendered in the A-RES modes cannot be displayed using this command.

If	RC	ADPRO_RESULT
an image was displayed	0	undefined
only A-RES mode Amiga displayable data is available	10	“Error”

13.11.4 Ending the Display of the Image

Syntax: **ADPRO_UNDISPLAY**

This command will move the screen containing any Amiga displayable image, which might be currently defined, to the backmost screen. The screen which will be revealed may not necessarily be the ADPro main screen. If this is required, you must issue an **ADPRO_TO_FRONT** as well.

Operation	RC	ADPRO_RESULT
always returns	0	undefined

13.11.5 Render Mode Status

Syntax: **RMODE** (1)
 RMODE (LOCKED | UNLOCKED) (2)

This command is equivalent to the **Locked Screen Mode** command under the **Settings** menu.

The first form gets the current status (**LOCKED** or **UNLOCKED**) of the render mode.

The second form sets the render mode status to the given value.

If operation was	RC	ADPRO_RESULT
successful	0	the <i>previous</i> render mode status
not successful	10	“Invalid Argument To Function”

13.11.6 Render Depth Status

Syntax: **RDEPTH** (1)
 RDEPTH (LOCKED | UNLOCKED) (2)

This command is equivalent to the **Locked Screen Depth** command under the **Settings** menu.

The first form gets the current status (**LOCKED** or **UNLOCKED**) of the render depth.

The second form sets the render depth status to the given value.

If operation was	RC	ADPRO_RESULT
successful	0	the <i>previous</i> render depth status
not successful	10	"Invalid Argument To Function"

13.11.7 Availability of Render Modes

Syntax: **AVAIL_MODES_ONLY** (1)
AVAIL_MODES_ONLY (ON | OFF) (2)

This command is equivalent to the **Available Modes only** command under the **Settings** menu.

The first form gets the current availability (**ON** for available only or **OFF** for all) of render modes.

The second form sets the availability of render modes to the given value.

If operation was	RC	ADPRO_RESULT
successful	0	the <i>previous</i> availability status
not successful	10	"Invalid Argument To Function"

13.11.8 Removing the Raw Image Data

Syntax: **CLEAR_RAW**

This command will remove the raw image data from ADPro image buffer.

Operation	RC	ADPRO_RESULT
always returns	0	undefined

13.11.9 Removing the Rendered Image Data

Syntax: **CLEAR_RENDERED**

This command will remove the rendered image data from ADPro image buffer.

Operation	RC	ADPRO_RESULT
always returns	0	undefined

Mode	Value
Hi Resolution On	1
Interlace On	2
PAL On	4
Horizontal Overscan On	8
Vertical Overscan On	16
VGA Mode On	32
Super Hi-Res Mode On	64
Super72 Mode On	128
Default Mode On	256

Table 13.4: Mask values for setting the screen type.

13.11.10 Closing the Render Screen

Syntax: `CLOSE_RENDER_SCREEN`

This command will close the render screen which was opened by the last **Re-display** or **Execute** command. This is as a convenience so you don't have to see the render screen when flipping through screens.

Operation	RC	ADPRO_RESULT
always returns	0	undefined

13.11.11 Screen Type

Syntax: `SCREEN_TYPE` (1)
`SCREEN_TYPE mask` (2)

The first form of the command returns the current screen type in `ADPRO_RESULT`. The value returned is in the form of a bit-mask comprised of the values indicated in Table 13.4.

The second form of the command allows the screen type to be set to the supplied value. The supplied value must be within the range of 0 to 31.

If value was	RC	ADPRO_RESULT
accepted	0	the <i>previous</i> screen type
out of range	10	"Invalid Argument To Function"

Your ARexx program might include the following:

```

HIRES = 1
INTERLACE = 2
PAL = 4
HOVERSCAN = 8
VOVERSCAN = 16
VGA = 32
SUPERHIRES = 64
SUPER72 = 128
DEFAULT = 256

```

Then, elsewhere in your ARexx program you can say things like:

```
SCREEN_TYPE HIRES + INTERLACE + HOVERSCAN + VOVERSCAN
```

This command would set the screen type to high resolution, NTSC, interlace, with both horizontal and vertical overscan.

Note that some modes cannot be used with some others. Specifically the flags for VGA, PAL, DEFAULT, and SUPER72 are mutually exclusive and the flags for Super Hi-Res and Hi-Res are mutually exclusive. If you specify these combinations, an error will result, which sets **ADPRO_RESULT** to **Invalid Argument To Function**.

Another possible error condition is setting a legal combination of flags which simply aren't supported on the Amiga you are currently using. For example, asking for Super Hi-Res on a machine without Super Hi-Res capability will fail. The error returned in **ADPRO_RESULT**, which indicates that the specified flags were legal but that ADPro could not comply, is simply **ERROR**.

13.11.12 Setting the Render Screen Mode

Syntax: **SET_RENDER_MODE** (1)
SET_RENDER_MODE *module_name screen_id width height colors* (2)

The first form returns the current mode information.

The second form sets the current screen mode to render to. The parameters that are given are the same as what is returned by the **GET_SCREEN_MODE** command.

This command is more than a combination of the **SCREEN_TYPE** and **RENDER_TYPE** commands, because it allows you to also specify all supported rendering options.

ID	Method
0	None
1	Floyd-Steinberg
2	Burkes
3	Sierra
4	Jarvis
5	Stucki
6	Random
7	Large Ordered
8	Small Ordered

Table 13.5: Dither methods and their identifiers.

If	RC	ADPRO_RESULT
successful	0	undefined
<i>width, height, or depth</i> is illegal	10	"Invalid Argument To Function"
render module is not available	10	"Library not found"

NOTE Other errors may be undetectable until rendering is requested.

13.11.13 Dither Selection

Syntax: **DITHER** (1)
DITHER *method_number* (2)

The first form returns the identifier of the currently selected dither method. The dither methods currently defined are given in Table 13.5.

The second form sets the dither method to the method corresponding to the supplied value, which must be in the range of 0 to 8.

If value was	RC	ADPRO_RESULT
accepted	0	the <i>previous</i> dither method
out of range	10	"Invalid Argument To Function"

See Section 5.6.3 for more information about dithering.

13.11.14 Dither Amount

Syntax: **DITHER_AMOUNT** *amt*

This command lets you specify the amount of effect for the Random and Ordered dithers. The *amt* parameter must be between 1 and 256, with 1 having the least effect and 256 having the most. If a dither other than Random or Ordered is specified, then this setting is effectively ignored.

13.11.15 Image Width

Syntax: **XSIZE**

This command returns the width of the currently loaded image in **ADPRO_RESULT**.

Operation	RC	ADPRO_RESULT
always returns	0	width of image

13.11.16 Image Height

Syntax: **YSIZE**

This command returns the height of the currently loaded image in **ADPRO_RESULT**.

Operation	RC	ADPRO_RESULT
always returns	0	height of image

13.11.17 Determining Currently Available Data

Syntax: **IMAGE_TYPE**

This command returns a string describing what kinds of data are available within ADPro at the present time. The string is returned in **ADPRO_RESULT** and can have the following forms:

NONE

This means that no image data (either raw or rendered) is currently available within ADPro.

COLOR

This means that raw color data is available but no rendered data is available at the current time.

GRAY

This means that raw grayscale data is available but no rendered data is available at the current time.

BITPLANE

This means that rendered image data is available but that no raw color or grayscale data is presently available.

COLOR BITPLANE

This means that both raw color and rendered image data are currently available.

GRAY BITPLANE

This means that both raw grayscale and rendered image data are currently available within ADPro.

Operation	RC	ADPRO_RESULT
always returns	0	NONE, COLOR, GRAY, BITPLANE, COLOR BITPLANE, or GRAY BITPLANE

The ARexx programmer must not make any assumptions about the order in which the words in the result string will appear. Instead of direct string comparisons to determine the contents of the result string, use ARexx's substring functions.

13.11.18 Pixel Aspect

Syntax: `PIXEL_ASPECT`

This command returns the current pixel aspect ratio in `ADPRO_RESULT`.

If an image was	RC	ADPRO_RESULT
available	0	pixel aspect
not available	10	unchanged

13.11.19 Pixel Resolution

Syntax: `PIXEL_RESOLUTION`

This command returns the current pixel resolution (horizontal and vertical DPI) in `ADPRO_RESULT`.

If an image was	RC	ADPRO_RESULT
available	0	pixel resolution
not available	10	unchanged

13.11.20 Amiga

The module name is “**Amiga**”. When used with ARexx commands which need a *screen_id* argument, replace *screen_id* with the mode id returned by the `GET_SCREEN_MODE` command.

13.11.21 EGS

The module name is “**EGS**”. When used with ARexx commands which need a *screen_id* argument, replace *screen_id* with the mode id returned by the `GET_SCREEN_MODE` command.

13.11.22 FC24

The module name is “**FC24**”. When used with ARexx commands which need a *screen_id* argument, replace *screen_id* with the mode id returned by the `GET_SCREEN_MODE` command.

13.11.23 OpalVision

The module name is “**OpalVision**”. When used with ARexx commands which need a *screen_id* argument, replace *screen_id* with the mode id returned by the `GET_SCREEN_MODE` command.

13.11.24 Picasso

The module name is “**Picasso**”. When used with ARexx commands which need a *screen_id* argument, replace *screen_id* with the mode id returned by the `GET_SCREEN_MODE` command.

13.11.25 Retina

The module name is “**Retina**”. When used with ARexx commands which need a *screen_id* argument, replace *screen_id* with the mode id returned by the `GET_SCREEN_MODE` command.

13.11.26 Window

The module name is “**Window**”. When used with ARexx commands which need a *screen_id* argument, replace *screen_id* with the public screen name to open this window on.

13.12 FRED Preferences

The ADPro program can be invoked by either double-clicking on its Workbench icon or typing its name at a Shell prompt. You can also customize various parts of the program by specifying some Tool Types or using a few command line arguments.

13.12.1 Public Screen Name

FRED's public screen name is "**Fred**". This cannot be changed.

13.12.2 Tool Type Options

You can specify Tool Type options (using FRED's icon) to control FRED. These include:

SHALLOW

If defined, the FRED screen will only open in 4 bit-planes on an AGA-equipped machine.

If not defined, the screen will open in it maximum depth (8 bit-planes in AGA, 4 bit-planes in non-AGA).

13.12.3 Command Line (Shell) Arguments

Alternatively, you can use one of more of the Command Line (Shell) options :

SHALLOW

If defined, the FRED screen will only open in 4 bit-planes on an AGA-equipped machine.

If not defined, the screen will open in it maximum depth (8 bit-planes in AGA, 4 bit-planes in non-AGA).

13.13 FRED Keyboard Qualifiers

Cell Selection	
Mouse/Keyboard Sequence	Function
single-click	Select a cell
double-click	Display a cell's image information
Shift + double-click	Load cell's image into ADPro

13.14 FRED Menus

This section gives a brief description of each menu item available in the FRED menu bar.

13.14.1 Project Menu

New... creates a new sequence window.

Open... displays a file requester, from which a previously-saved sequence can be loaded.

Save stores the current sequence's changes under the sequence filename.

Save As... stores the current sequence's changes under a possibly-different sequence filename. A file requester is displayed to specify a filename.

Reveal... displays a list of loaded sequences, from which one can be opened and activated.

Close... hides the sequence window, but does not unload the sequence information.

About... displays information about the program.

Quit FRED exits the program,

13.14.2 Settings Menu

Text Options... displays the Text Options control panel, from which the text underneath each cell can be selected.

Load Settings... displays a file requester, from which a previously-saved FRED settings file can be opened and used.

Save Settings stores the current program settings under the default settings filename (**FRED.prefs**).

Save Settings As... displays a file requester, from which a filename can be specified for the current settings.

13.14.3 Tools Menu

ADPro / MorphPlus activates either ADPro or MorphPlus, whichever it finds first. It looks for these applications in the **ADPRO**:-assigned directory, as

well as in the **SYS:ADPro** and **SYS:MorphPlus** subdirectories, respectively.

Morph activates the Morph program. It looks for this application in the **ADPRO:-assigned** directory, as well as in the **SYS:Morph** subdirectory.

Workbench brings the Workbench screen to front.

13.14.4 AnimOps Menu

This menu contains menu items for each **AnimOp** found in the **ADP_FRED:AnimOps** subdirectory.

13.14.5 Edit Menu

This menu is only available when a sequence window is active.

The **Insert** submenu contains commands for inserting images into the current sequence window. **Image(s)...** lets you insert one or more images from the same directory. **Range...** lets you insert a range of images (it looks for numbers at the end of the filenames). **Directory...** lets you insert all the files within a particular directory.

Cut removes the selected cells, placing them into the Clipboard.

Copy duplicates the selected cells to the Clipboard.

Paste inserts the cells currently in the Clipboard into the sequence in the order that they appear.

Reverse Paste performs a paste operation, but inserts them in reverse order that they appear in the Clipboard.

Select All selects all the cells in the sequence.

Select Range selects all the cells from the currently active cell to the next cell you click.

Deselect Range deselects all the cells from the currently active cell to the next cell you click.

Duration... displays the Cell Duration control panel, from which the length of all selected cells can be set.

Information... displays the Image Info control panel, from which the currently active cell's length can be set.

13.14.6 Animation Menu

This menu is only available when a sequence window is active.

The **Playback** submenu contains controls for playing the cell's stamps in an animation. **Once...** plays them only once from start to finish. **Continuous...** plays them in loop mode. **Ping Pong...** toggles between forward and backward playback.

The **Range** submenu contains controls for specifying which of the cells to animate. **All** will play all cells. **Selected** will only play the selected cells.

The **Speed** submenu contains controls for playback speed. These will show differently if running on an NTSC (**30**, **15**, **12**, and **10**) versus PAL (**25**, **15**, **12.5**, and **10**) machine.

Make Stamp... uses ADPro or MorphPlus to create a stamp-size image of the currently selected cells for display within the cell.

13.14.7 Scripts Menu

This menu is only available when a sequence window is active.

Process... displays the Invoke ADPro control panel, from which a sequence of ARexx scripts (a.k.a., FREDScripts) can be performed on the selected cells in the sequence.

Appendix A

Troubleshooting

A.1 Technical Support

ASDG provides technical support and can accept your questions or comments via the methods described in Table A.1.

ASDG representatives occasionally stop in on Usenet as well.

NOTE Please remember that technical support cannot be rendered to anyone who is not in our registered user database.

Also, please fill out the following form and have it available when you call ASDG for technical support.

Access	Type	Information
Business Address	Surface Mail	ASDG, Incorporated 925 Stewart Street Madison, WI 53713 USA
General Info	Telephone	(608) 273-6585
Tech Support	Telephone	(608) 273-6585
Direct Sales	Telephone	(608) 273-9240
Internet	E-Mail	pk-asdg@cup.portal.com asdg2@bix.com
Portal	Support Area E-Mail	go asdg pk-asdg
BIX	Support Area E-Mail	join asdg asdg2
CompuServe	Support Area E-Mail	go amigav, Section 2 76004,1765
GEnie	Support Area E-Mail	move 555;1, Category 27 ASDG.TECH

Table A.1: Methods of Contacting ASDG.

Product Name:
Product Version:
Product Serial Number:
Computer Model / Processor:
Kickstart / Workbench Version:
Fast / Chip (Graphics) Ram:

Peripheral #1

Type:
Model:
Port Connection:
Driver:
Driver Version:

Peripheral #2

Type:
Model:
Port Connection:
Driver:
Driver Version:

Peripheral #3

Type:
Model:
Port Connection:
Driver:
Driver Version:

Peripheral #4

Type:
Model:
Port Connection:
Driver:
Driver Version:

Appendix B

Hints and Tips

This appendix lists some hints and tips you might find helpful when working with ADPro. If you find other hints and tips while working with the programs that come with this package and would like to share these with other ADPro users, you are more than welcome to send us a letter, call us on the phone, or leave electronic mail at any one of the various electronic information networks of which an ASDG representative is a member. If we find that your hint or tip may be helpful to other users, we will include it in future versions of the manual and post it on these information networks.

B.1 ADPro

The topics discussed in this section include:

- Anti-Aliasing
- Approximating Charcoal Drawings
- Avoiding the Creation of Corrupt IFF Files
- Caching the Loader, Saver, and Operator Modules
- Changing Input Field Values
- Compositing Tips
- Creating Better Gradated Fills
- Creating Drop Shadows
- Eliminating Stray Dots
- Example FG and FC24 Use

- Focused Color Picking
- Grayscale Balancing
- Merging a Color Image Into a Grayscale Image
- Mixing Picked and Computed Colors
- Multiple ADPro Configurations from the Workbench
- Parsing Strings in ARexx
- Reading and Writing Toaster Files
- Regionalized Processing
- Scaling as an Anti-Aliasing Tool
- Starting ADPro Via ARexx
- Suggested PostScript® Parameters For Color Work
- Time Lapse and Motion Blur
- Turning Low Color Images Into Higher Res Grayscale
- Working With Moving Video
- Working With Pictures Which Are Too Large

B.1.1 Anti-Aliasing

The jaggie edges that often detract from computer generated images are called aliasing problems. The term is derived from the fact that computers divide images into little color boxes called pixels. A pixel, being of a given shape and having a well defined border with its neighboring pixels, cannot perfectly model the color information it represents. The error between what a pixel can represent and what it is supposed to represent causes the jaggies (or aliases).

It is common to want to eliminate the jaggies from your images. This process is called anti-aliasing. Note that when you anti-alias (or remove the jaggies) you must, in effect, blur the original image to some degree. In other words, to eliminate the jaggies, you must give up some amount of spatial resolution. ADPro has several tools to anti-alias your images and gives you control over how much actually blurring will take place.

B.1.2 Approximating Charcoal Drawings

You can create a charcoal-like result by combining some of the features of ADPro. If you are starting from a color image, turn it into grayscale (using the Color_To.Gray operator). Boost the contrast heavily and select the Line_Art operator. Next, select a 2 color screen and Floyd-Steinberg dithering and render the image. Finally, execute the Rem_Isolated_Pxls operator. The resulting image should have a rubbed charcoal-like quality.

B.1.3 Avoiding the Creation of Corrupt IFF Files

If you find that you are getting “Mangled IFF” messages from ADPro after saving or loading IFF files, you might want to suspect that the MaxTransfer value for your hard drive is set too high. The default is `0xffffff`. You might want to lower it to `0xffff` and see if that fixes them.

B.1.4 Caching the Loader, Saver, and Operator Modules

ADPro can cache loaders, savers, and operators in a higher speed disk (such as **RAM**:). To make use of this feature, create an environment variable called **ADPRODIR** and set it to **RAM**:. Then create **Loaders2**, **Savers2**, and **Operators2** directories. Copy into these directories the loaders, savers, or operators you will use most often.

These LSOs (Loader, Saver, and Operators) will then load from **RAM**: instead of from disk, resulting in faster performance (especially during massive batch processes).

For example, you might make a Shell script such as this (or perhaps include something like this in your user-startup):

Assume you want to cache:

- IFF and JPEG loaders
- IFF and TEMP saver
- Crop_Visual operator

You would execute the following commands:

```
SetEnv ADPRODIR RAM:
MakeDir RAM:Loaders2
MakeDir RAM:Savers2
MakeDir RAM:Operators2
Copy ADPRO:Loaders2/(IFF|JPEG) RAM:Loaders2
Copy ADPRO:Savers2/(IFF|TEMP) RAM:Savers2
Copy ADPRO:Operators2/Crop_Visual RAM:Operators2
```

This mechanism also provides a way of splitting modules across multiple volumes (like floppies). To make ADPro run from multiple floppies (for example), create a primary floppy disk with as much of ADPro as can fit. Include **empty files** for all the modules which don't fit. If you don't include the empty dummy files, ADPro won't know about these modules.

Put the real versions of the modules which wouldn't fit on the second floppy in the appropriate directories and execute the appropriate **SetEnv** command.

B.1.5 Changing Input Field Values

Just a reminder, whenever you change the value of an input field, you **MUST** press either the **Return** or **Tab** key to acknowledge the change to the program. Otherwise, the program will not know you have changed the value, even though the value is displayed in the input field.

B.1.6 Compositing Tips

ADPro's image compositing capabilities include the ability to select a specific color as being transparent. As an image is loaded during a compositing operation, each pixel is compared against the transparent color. If the pixel matches the transparent color, the new pixel is ignored and the pixel from the old image is left untouched.

This capability can be used in a two step masking process to paste an arbitrary shape from one image into another.

Suppose you had a 24 bit-plane scan of a full moon that you wish to place into a ray-tracing.

First, load the scan of the moon into ADPro. Render the image (with dithering) in any Amiga display mode which is compatible with your favorite paint program. We recommend 32 colors, low resolution, non-interlaced. We suggest that this rendering be made with dithering on so that you can get the best impression of what the 24 bit-plane data looks like.

Save this rendering under a different name from the original 24 bit-plane data. Do not overwrite the 24 bit-plane data as this will be needed later.

Load the rendered data into your favorite paint program. For our example, we assume Deluxe Paint IV. Bring up the image's palette. Make color register 0 completely black, that is, make its red, green, and blue components all zero. Also, make the highest color register completely white (red, green, and blue all 15's).

Select color register 0 (completely black) as your pen color and trace around the perimeter of the moon. When this is done, fill the outside of the moon completely with color register 0. In Deluxe Paint IV, this can be done easily with **Alt-Fill**. Next, fill the inside of the moon with the completely white color register. Save this image (overwriting its former self is acceptable). This is your mask.

Load the original 24 bit-plane moon image. Select the image compositing mode and then load the mask. When the image compositing control panel comes up, specify that you wish white (255, 255, 255) to be the transparent color. Specify an *x* and *y* offset of 0 so that the images completely overlap. Perform the load.

What you should now have in memory is the 24 bit-plane moon left completely untouched. However, its background should now be set completely and uniformly to black. Save this 24 bit-plane file for use later.

To complete the process load the ray-tracing. Then, in compositing mode, load the 24 bit-plane image saved in the previous step. This time, specify black (0,0,0) to be the transparent color. This instructs ADPro to load just the moon into the ray-tracing since the background of that image is uniformly “transparent”.

B.1.7 Creating Better Gradated Fills

For display upon the Amiga, here are some tips for creating smoother gradated fills. If you are using the actual 24 or 8 bit-plane data (for example, driving a 24 bit-plane film recorder or display board) then these tips are not as useful.

- Where possible, vary only one primary color.
If you vary only one primary color in a gradated fill, the resulting fill will appear much smoother than if more than one primary color varies. This is because of the limited color resolution of the Amiga. If you vary more than one primary color, the primaries will vary at different rates causing a strobing effect in the Amiga displayable image.
- Where possible, draw in an oversized bitmap.
For example, if you wish to produce a 320 by 200 end product, you will get better results by creating a 640 by 400 bit-map and then scaling downward by 50 percent. The larger the size of the bitmap, the more finely ADPro will create a gradated fill.

B.1.8 Creating Drop Shadows

Drop shadows can be created using the Rectangle or Rectangle_Visual operators and by using a combination of mixing and transparency during image compositing. To create a drop shadow for a rectangular region, simply draw a filled black rectangle at a 50 percent mix. To create a drop shadow for a complex image, create a black and white mask where the black area is the area which will cast the shadow. Then, use image compositing to load in the mask at an offset and at a 50 percent mix with white being transparent.

After the shadow has been created, load in the image which is supposed to cast the shadow. Since this image is loaded after the shadow is drawn, it will obscure the appropriate areas to make a realistic looking shadow.

B.1.9 Eliminating Stray Dots

As mentioned elsewhere in this manual, there are two principal methods for eliminating stray dots which can crop up especially in dithered renderings. These are:

1. Increasing the contrast of the image very slightly will often eliminate huge amounts of stray dots especially in dithered renderings. Increasing the contrast has the effect of decreasing the subtle variations in shading which the dithering routines are trying to represent when they produce seemingly stray dots.
2. The RIP (RemIsolatedPxls) function can also remove stray dots very effectively. This is in fact, the purpose for which it was designed. RIP is especially useful when working with line or drawn art.

However, when working with scanned art be aware that applying RIP will remove isolated pixels in the entire image. This means that RIP will slightly degrade the quality of whatever dithering was used by removing some stray dots that were visually important.

Therefore, when working with full color images we suggest that you first attempt to deal with stray dots by slightly increasing the contrast of the image before using RIP.

B.1.10 Example FG and FC24 Use

The ARexx program below is an example which shows how the FRAMEGRABBER loader and the FC24 saver might be used. This program also contains an example usage of the **VERSION** command.

NOTE The FRAMEGRABBER loader does not operate correctly on PAL machine.

```
/*  
** Short program showing a sample of how the  
** FRAMEGRABBER loader and the FIRECRACKER  
** saver are used.  
*/
```

```
ADDRESS "ADPro"
```

OPTIONS RESULTS

```
ADPRO_TO_FRONT
ORIENTATION "PORTRAIT"
```

```
LFORMAT "FRAMEGRABBER"
IF (RC ~= 0) THEN DO
    ADPRO_TO_FRONT
    OKAY1 "Could Not Select The FrameGrabber"
    EXIT 10
END
```

```
SFORMAT "FC24"
IF (RC ~= 0) THEN DO
    ADPRO_TO_FRONT
    OKAY1 "Could Not Select The FireCracker"
    EXIT 10
END
```

```
/*
** The following two lines which init the
** FireCracker could have been done in a
** single command. They have been broken
** down into several commands purely to
** make each line fit in the manual.
*/
```

```
SAVE "XXX" "RAW" "BOARD_NO" 0 "B_WIDTH" 768
SAVE "XXX" "RAW" "CLEAR" "BOARD" "ON"
```

```
/*
** Note example of ADPro version checking.
*/
```

```
VERSION
IF (WORD( ADPRO_RESULT, 2 ) > "1.0.2") THEN DO
    ADPRO_TO_FRONT
    OKAY1 "Select An Abort Button When Done"
    Counter = -1
END
ELSE DO
    ADPRO_TO_FRONT
    OKAY1 "This Will Load/Save 10 Times"
    Counter = 9
END
```

```

DO WHILE (Counter ~= 0)
    LOAD "XXX" "FIELD1"
    IF (RC ~= 0) THEN
        EXIT 10

    SAVE "XXX" "RAW" "CENTER" "IMAGE"
    IF (RC ~= 0) THEN
        EXIT 10

    Counter = Counter - 1
END

```

B.1.11 Focused Color Picking

Sometimes, when rendering a 24 bit-image into a small number of colors, you might want to channel ADPro's color picking to give a small area of the total bit-map the best possible color choices (while sacrificing color quality elsewhere in the image).

Left alone, ADPro will choose the color which best matches the entire image. However, using the Crop_Visual operator, you can channel ADPro's color picking to optimally pick colors based upon only a small subsection of the entire image.

This can be done as follows:

1. Save out the 24 bit-plane image because you will need to preserve it through the next steps.
2. Using the Crop_Visual operator, crop the area of focus for color picking.
3. Have ADPro choose the desired number of colors based upon the cropped area.
4. Lock the palette.
5. Reload the original image saved in the first step. Remember, the palette is locked.
6. Render the entire image in the desired number of colors using the locked palette. Because you are using the locked palette based only upon the sub-region, that region will have optimal color choices within the entire image.

B.1.12 Grayscale Balancing

We have found that some very striking results can be had in grayscale by heavily increasing contrast and then adjusting brightness or gamma or both.

While color images are best brightened using the gamma correction, we have found that adjusting the brightness of a grayscale image (using the Brightness control) will give a more natural looking result.

B.1.13 Merging a Color Image Into a Grayscale Image

When merging a grayscale image into a color background, you are adding an 8 bit-plane image to a 24 bit-plane background. However, to add a color image to a grayscale background, you would need to add a 24 bit-plane image to an 8 bit-plane background. Since 24 bit-planes won't fit in 8, but the ability to add a color image to a grayscale background would be nice, the `Gray_To_Color` operator was defined.

It takes 8 bit-plane raw image data and creates the equivalent grayscale data spread over 24 bit-planes. It does this by making each primary color equal and, thereby, gray.

To add a color image to a grayscale background you would first load the grayscale image which will become your background. Execute the `Gray_To_Color` operator. This causes ADPro to create 24 bit-plane data (which happens to be gray) from the original 8 bit-plane data.

Now, you can directly merge a color image since 24 bit-planes are now available for ADPro's internal use.

B.1.14 Mixing Picked and Computed Colors

In some instances it might be desired to pick some of the screen's colors by hand and let the computer pick the rest. For example, you might want to ensure that several specific colors are present in the rendered image.

Let's take a complicated example of a 32 color screen in which the following requirements are to be met:

- Color registers 0 through 7 are to be ignored completely.
- Color registers 8 through 15 will contain hand picked colors.
- Color registers 16 through 31 are to be picked by the computer.

This can be accomplished as follows:

1. Enter the Set Render Screen requester.

2. Choose a native Amiga display mode and set the **Colors:** value to 32. Select the **OK** button when done.
3. Enter the Palette control panel.
4. Select the **Custom Palette** checkbox so that it is “on”.
5. Set **Colors (Used):** to 16 and **Offset Color Zero:** to 16.
6. Select the **OK** button in the Palette control panel.
7. Select the **Execute** button.

The resulting image will have 16 computer picked colors in registers 16 through 31.
8. Next, enter the Palette control panel again.
9. Set **Colors (Used):** to 24 and set **Offset Color Zero:** to 8.
10. Edit the palette to place the hand picked colors into registers 8 through 15 (this can be accomplished via palette loading as well).
11. Set the palette to **Locked** and exit the Palette control panel.
12. Finally, rerender the image with the **Custom Palette** checkbox still selected.

The resulting image will use all registers between 8 and 31 including the 8 which were hand picked and the 16 which were computer picked. It will avoid color registers 0 through 7 completely.

Using this technique you can have enormous flexibility in color composition for video titling, animation, or other video applications.

B.1.15 Multiple ADPro Configurations From The Workbench

To start ADPro with custom configurations, you can create “dummy” project icons that use ADPro as its Default Tool, but which only differ in the Tool Types specified. This will allow you to customize the ADPro environment.

For example, you can have one icon that launches ADPro with a maximum image buffer of 5,000,000 bytes (**MAXMEM=5000000**), and another one that uses up the largest contiguous chunk (no Tool Type required). Another icon can specify normal palette accuracy mode (**NOENHANCED**), and another can load a custom file of settings (**SETTINGS=Work:Miscellaneous/ADPro1.prefs**).

For more information on Tool Type options, please read Section 13.1.

B.1.16 Parsing Strings in ARexx

As a helpful tip for ARexx script writers, if you want to parse a sequence of quoted string (which the **GETLIST** and **LISTVIEW** ARexx commands return in **ADPRO_RESULT**), you can use the following technique:

Assuming your sequence is in a variable called **StringList**, to parse the first quoted string in the sequence, you can use the following command:

```
PARSE VAR StringList '''FirstString''' RestOfString
```

After this call, the **FirstString** variable will contain the first quoted string and **RestOfString** will contain the rest of the line.

B.1.17 Reading and Writing Toaster Files

NewTek's Video Toaster software can read and write 24 bit-plane IFF files. To do so, enter Toaster Paint and select the loading or saving of RGB files. The RGB format is precisely 24 bit-plane IFF.

Note that NewTek's software does not understand the color balancing chunks (**CLUT** chunks) placed in 24 bit-plane IFF files by many program, such as ADPro. See Section 8.2 for a discussion of the **Apply_Map** operator which transfers the information that would have been contained in the **CLUT** chunks to the image data itself.

Also see Section 8.16 for information about how to convert grayscale image data into color image data as the Toaster software cannot make use of 8 bit-plane grayscale IFF files.

Alternatively, you can use the more convenient (i.e., you do not need to use Toaster Paint) **FRAMESTORE** loader and saver modules to transfer images from and to Framestore format.

B.1.18 Regionalized Processing

In addition to basic compositing, alpha channels can be used to apply one of ADPro's operators to a region of an image. Here is how you would do it:

1. Use a paint program to create a black and white image, where the white areas represent the regions of the image to be processed.
2. Save this image as your alpha channel or matte file.
3. Load in the image you want to process.

4. Save it to the TEMP buffer, if you have enough memory, for faster retrieval later in these instructions.
5. Process your image with one of the image operators.
6. Save this image to a temporary file as your modified image.
7. Reload the original image.
8. Use the instructions in the manual (Section 5.3.3) for compositing an image using an external alpha channel.

Voila! The operation you performed has only affected the region you defined by the matte.

B.1.19 Scaling as an Anti-Aliasing Tool

ADPro's scaling technology employs a sophisticated bilinear interpolation engine with 64 bits of precision. In plain talk, ADPro's scaling is extremely good. And it can be used to anti-alias your images very simply.

If you have a large image that you need to scale down to a smaller size, simply scaling the image to the desired size will also anti-alias the image with optimal results.

On the other hand, if your image is already the appropriate size, you can scale it to, perhaps, 99 percent of its original size, and then scale it back to its original size. The resulting image will be nicely anti-aliased with minimal loss.

B.1.20 Starting ADPro Via ARexx

The following ARexx fragment can be adapted for use in your own ARexx programs. This code attempts to locate a currently running ADPro. Failing that, it will attempt to start up ADPro. In order to do this, ADPRO: must be defined.

```
/*
** Locate-ADPro.rexx
**
** $VER: Locate-ADPro.rexx 1.0.1 (1.2.93)
**
** This ARexx program will attempt to find a currently
** running ADPro. If one is not found, then it will
** attempt to start up ADPro.
**
** The main guts of this program are embedded in a
```



```

** sub-routine to make it easier to glue into your own code.
**
** Example ARexx program for controlling ADPro.
**
** Copyright 1990-1993 ASDG, Incorporated
** All Rights Reserved
*/

```

```

OPTIONS RESULTS

```

```

CALL Locate_ADPro

```

```

IF (RESULT = 1) THEN
    SAY "ADPro has been found"
ELSE
    SAY "Could not locate or start ADPro"
EXIT

```

```

Locate_ADPro:

```

```

Max_Seconds_To_Load = 60
Flag = 0
LibName = 'rexxsupport.library'

```

```

IF (POS( LibName, SHOW( 'Libraries' ) ) = 0) THEN
    ADDLIB( LibName, 0, -30, 0 )
IF (POS( LibName, SHOW( 'Libraries' ) ) = 0) THEN
    RETURN 0

```

```

IF (STATEF( 'ADPRO:' ) = "") THEN
    RETURN 0

```

```

TIME( 'R' )
DO WHILE ((TIME( 'E' ) < Max_Seconds_To_Load) &
    (POS( 'ADPro', SHOW( 'Ports' ) ) = 0))
    IF (Flag = 0) THEN DO
        ADDRESS COMMAND 'Run < nil: > nil: ADPRO:ADPro'
        Flag = 1
    END
    ADDRESS COMMAND 'Wait 1'
END
IF (POS( 'ADPro', SHOW( 'Ports' ) ) = 0) THEN
    RETURN 0
ELSE
    RETURN 1

```

Plane	Density	Angle
Black	149.7	45.0
Cyan	133.9	108.4
Magenta	133.9	161.6
Yellow	141.1	90.0

Table B.1: Suggested angles and densities for color work run on the Linotronic L300 or L330.

Image Number	Mix Percentage	Contribution of Each Image
1	-	-
2	50	1/2
3	33	1/3
4	25	1/4
5	20	1/5
n	$100/n$	$1/n$

Table B.2: Mix percentages to produce a time lapse effect.

B.1.21 Suggested PostScript Parameters For Color Work

If you are running color films on a Linotronic L300 or L330, Table B.1 contains the angles and densities we suggest. We run our own color film with these values at 2540 DPI. Note that some new versions of PostScript® automatically alter your choices of angles and densities (without your knowledge). You should ask your service bureau if their version of PostScript® does this. If so, ask them to either disable this feature for your job (if you use our suggested angles) or have them suggest new angles which work properly with their version of PostScript.®

B.1.22 Time Lapse and Motion Blur

ADPro's compositing ability with variable mixing can be used to create a time lapse photography effect when working with multiple frames of an animation. This is done by mixing multiple frames together in a way that causes each individual frame to contribute equally to the final image.

Table B.2 indicates the mix percentage to be used when adding each additional frame to the time lapse.

By varying the mix percentages so that the frame being added contributes the most to the resulting image, you can achieve a motion blur effect.

B.1.23 Turning Low Color Images Into Higher Res Grayscale

You can take a 16 color Amiga format image (for example) and turn it into a grayscale image with many more than 16 shades of gray.

Load the low color image into ADPro. The IFF Loader will detect that the image is colored and will pad it into 24 bit-planes. Reduce the width and height of the image by more than 50 percent, then convert the image to grayscale. The resulting 8 bit-plane image will possess more than 16 gray shades.

B.1.24 Working With Moving Video

If you intend to use frames of moving video with ADPro, you should understand a few things.

First, as each frame of moving video is made up of two fields (an odd and even field, interlaced), once you modify the contents of the frame—with one of the ADPro operators—you are destroying this relationship between fields.

To work around this, you can use the Interlace and DeInterlace operators to go from frame to fields and back.

B.1.25 Working With Pictures Which Are Too Large

As you know, ADPro requires ($width * height * 4$) bytes of memory to process a color image. If you do not have enough memory to work on a particular image, there is a way you can work on the image in sections.

Using the BACKDROP loader, create a backdrop as large as the amount of memory on your machine will allow. Then, use the compositing capabilities of ADPro to load in sections of the huge picture which completely replace the backdrop.

By way of a concrete example, suppose you can only accommodate a 640 by 400 image and you wish to work on a 1280 by 800 image. Perform the following steps:

1. Select the BACKDROP loader. Load a backdrop measuring 640 by 400.
2. Select the appropriate loader for the huge file you wish to work on.
3. Make sure that compositing is enabled.
4. Load the image. When the compositing control panel appears, select an offset of (0,0), a mix percentage of 100 percent, and disable transparency. Accept these settings.

You just loaded the upper left hand corner of the image. You can repeat these steps to work on the other sections of the image by entering the appropriate x and y offsets in the compositing control panel.

NOTE While this method allows you to load in a large image in sections, there is no method for joining these sections together again into a single image. Therefore, this method is most useful in situations where you wish to extract a portion of a larger image for processing or where you wish to view a larger image.

Appendix C

Third-Party Developer Support

Third-party software developers or interested individuals wishing to create loader, saver, operator, or display modules for ADPro can apply for developer status.

As an ADPro developer, you will receive example source code for writing a module for ADPro or MorphPlus, header files, printed documentation, free technical assistance and developer kit upgrades.

Modules that you develop can be used for personal use, given away freely, or sold as part of a commercial package. There are specific quality assurance requirements which must be met for any modules intended for public distribution. Additionally, arrangements can be discussed to include these modules in future versions of the ADPro and MorphPlus packages for greater exposure.

There is a one-time application fee. Please call the ADPro Developer Program Administrator at our business phone number for more information, or write to the following:

ASDG, Incorporated
ATTN: ADPro Developer Program
925 Stewart Street
Madison, WI 53713
USA

Appendix D

PREFPRINTER Dither Type Samples

This appendix contains printed samples of the various dither types available with the PREFPRINTER saver.

For the **Ordered**, **Horizontal**, **Vertical**, **Fwd Brick**, **Bck Brick**, **Fwd Diagonal**, **Bck Diagonal**, and **Halftone A** and **B** dithers, each sample is divided into four sections. This allows you to compare the four different **Mask Size** values on the same image at the same time; the actual **Mask Size** used is labelled within the sections on the image.

Note that the **Halftone A** and **Halftone B** dithers are combined into one sample because their grayscale representations are identical. Color images may show a difference, however.

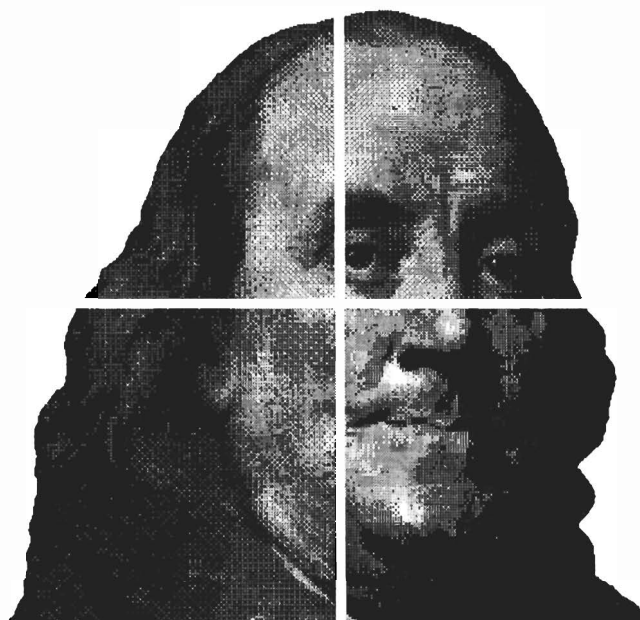
NOTE Although you are encouraged to examine these samples as a reference and as a “sampler” of what each dither will produce, you should be aware that each printer will produce slightly different output. Factors, such as type of paper and darkness of the ink or ribbon, might cause your actual prints to be different than what is shown here.



Floyd-Steinberg

16

4



8

2

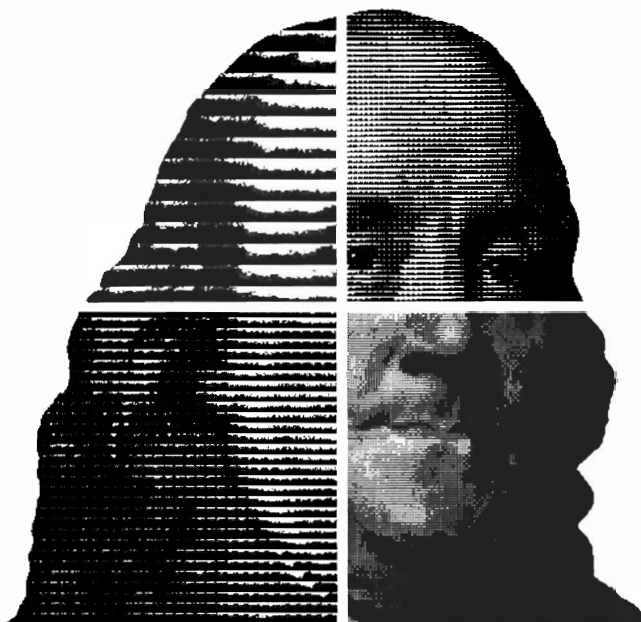
Ordered

16

4

8

2



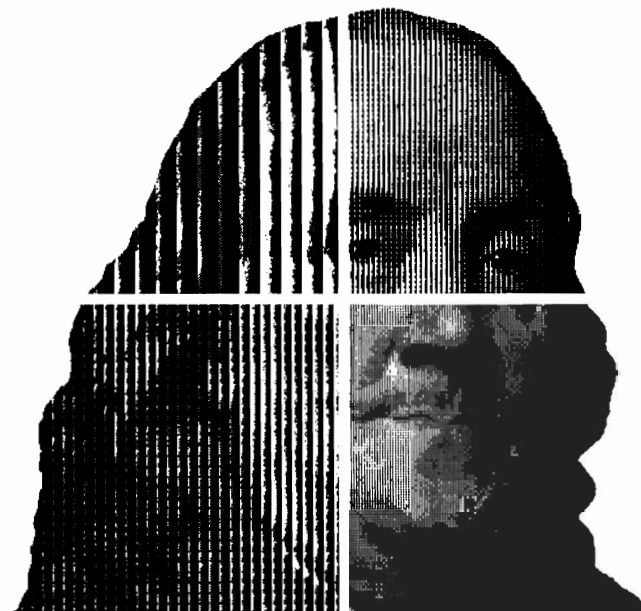
Horizontal

16

4

8

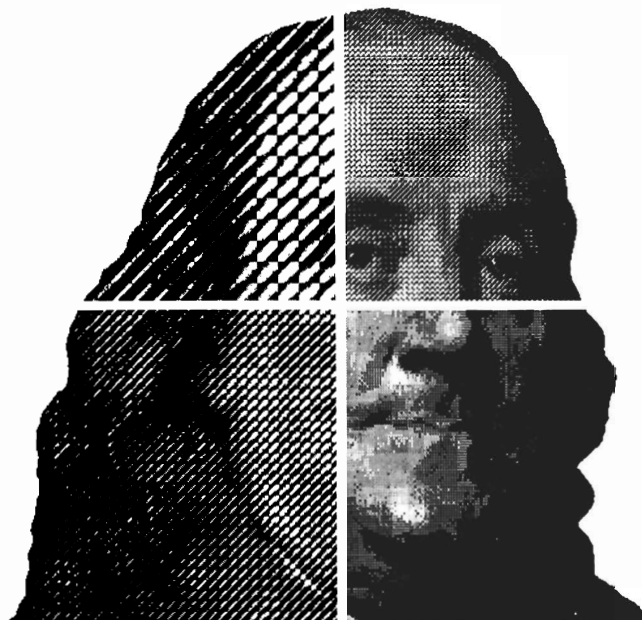
2



Vertical

16

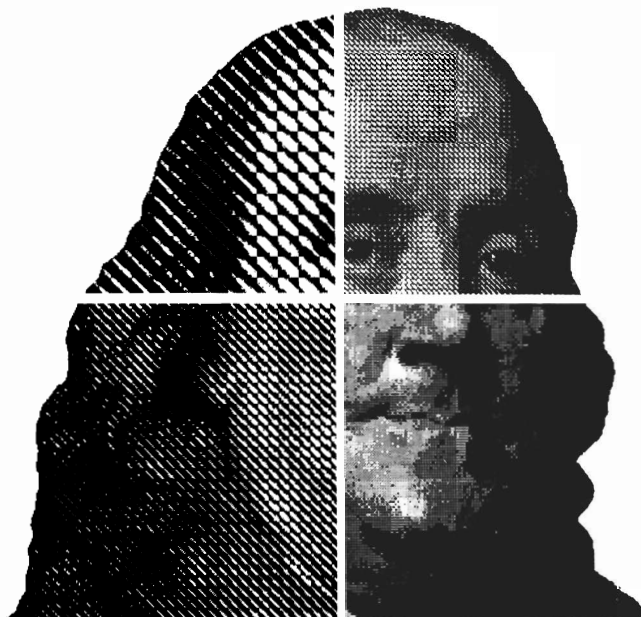
4



Forward Brick

16

4



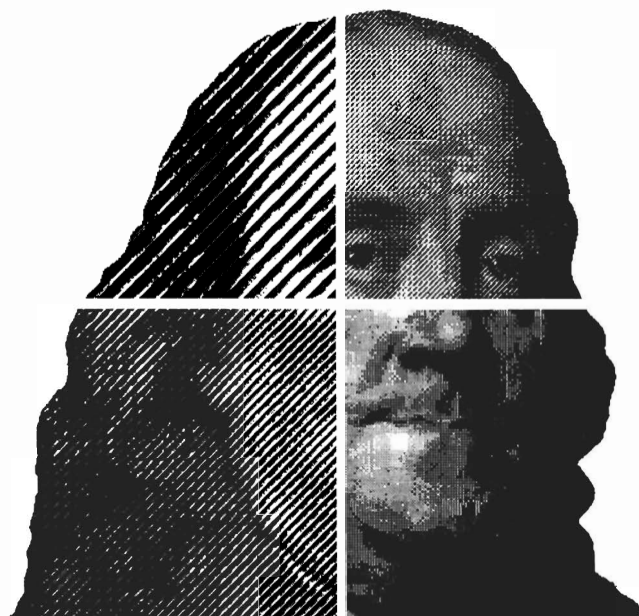
Backward Brick

16

4

8

2



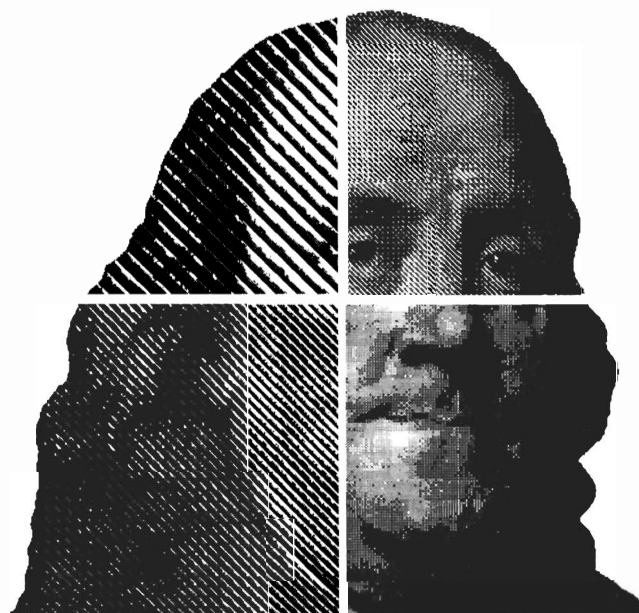
Forward Diagonal

16

4

8

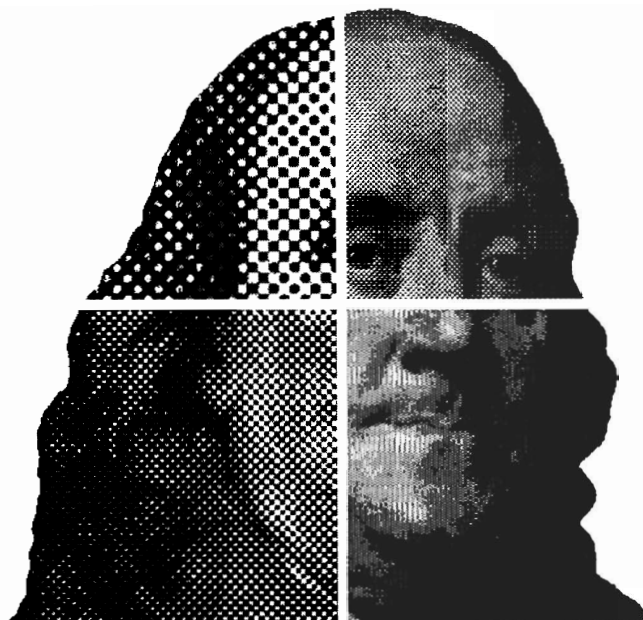
2



Backward Diagonal

16

4



8

2

Halftone

```

--+++-
-=M0ZMN0SSNWZ=-
;WXXWI++==+*IS0X+
-MH0HN+==:==+*I=W00
-000000*==:==+*II=MH0S
000000++==:==+I==I=NH00=
=000000++*SNNO==I=OMX000
+0000000++=WZ000N*=X000000:
-"0000000I=*ON0X0Z"*0000000
-0000000000H*==:==+0++=HWWW0000=
-H00000000000I++==+==+==SWS0W000*
=00000000000NI*++++==+IMMNH0NSW0000K-
-M0000000000HO=*+++=+++=0000Z0000000*
-X000000000000H0*++++*=SII=S2HM00000000-
+00000000000000Z=II+=+++=+I=0W0000000*
:0000000000000000I=I*+===*=00000000Z-
=000000000000000000=*==+I*++I=0W00000000-
000000000000000000X000000000000000000-
H00000000000000000000000000000000000S-
=000000000000000000000000000000000000M

```

ASCII

Appendix E

Additional Utilities

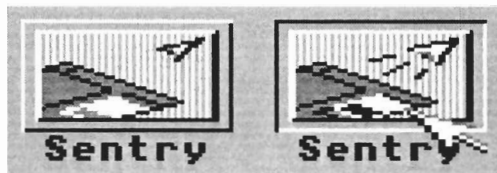
This appendix describes the utilities which are included on the Art Department Professional disks, but which are not required for use with the ADPro program.

E.1 Sentry

The Sentry utility allows you to monitor a directory, and execute a series of ARexx script or CLI commands on any new or updated files within the directory. This ability to modify files as they are created, such as converting frames generated by a 3D modeling program into DCTV images, not only saves hard drive space (by automatically deleting the original frames after they are modified), but also allows you to easily batch process files.

NOTE Sentry requires AmigaOS 2.04 or later to run.

E.1.1 Starting Sentry



For Workbench users, open the directory where the Sentry program is located. Locate this icon as shown in the left image above. It should have the name

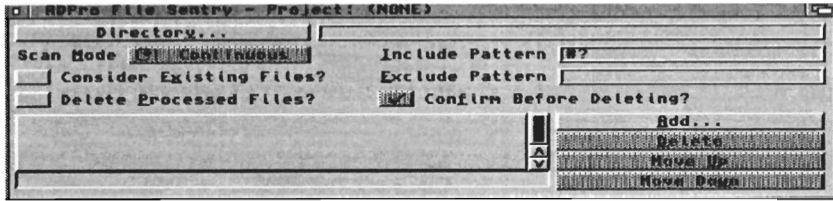


Figure E.1: Sentry's main control panel.

Sentry below it. Double-click on this icon to start the Sentry program.

For Shell users, change the current directory to the directory where Sentry is located and start the program with the two commands below. Type each command on its own line, hitting the **Return** key after each line.

```
cd Sentry-directory
Sentry
```

Sentry-directory must be replaced with the actual directory name where the Sentry program and its files were installed. Note that you do not need to prepend the **run** command before **Sentry** above—Sentry detaches itself and sets its own stack size of 16,000 bytes.

NOTE Sentry does not use the parent Shell's output, so you still need to specify the **OUTPUT** argument to get output from the program.

If Sentry is successful in initializing itself, then you will see the main Sentry control panel (shown in Figure E.1). If no panel is displayed, you might have run out of available memory for the program to use.

E.1.2 Configuring Sentry

The Sentry work environment can be customized to your needs by using the program icon's Tool Types.

The following attributes can be modified from Sentry's icon Tool Types list:

HOTKEY=*key_sequence*

This option allows you to specify a key sequence that, when invoked, will display the program's status control panel. This sequence will only work if the program is in directory scanning mode (it is hidden from view). To hide the status control panel, select the **Hide** button or menu item. The program now becomes a commodity.

The *key_sequence* follows standard Commodities syntax. If this Tool Type is not defined, the default *key_sequence* is **Ctrl + Alt + s**.

OUTPUT=*file_or_device*

This option allows you to specify a file or device to which output from the executed CLI / Shell programs will be directed. A example device could be something like **CON:**, which will open a small window for all program output.

If this Tool Type is not defined, the default **OUTPUT** is **NIL**: (no output displayed).

PROJECT=*project_filename*

This option allows you to load a previously saved project on startup.

HIDE=1

This option allows you to automatically hide Sentry once monitoring begins.

AUTOSTART=1

This option allows you to begin monitoring a directory on startup. As such, the **PROJECT** tool type must also be defined.

CX_PRIORITY=*level*

This option allows you to specify the commodity's priority in the same way you control the priority of other commodities. This value can be between -128 and 127. Setting *level* to a high number sets the commodity to a high priority. This allows for input events to be sent to this commodity before commodities at lower priorities.

If this Tool Type is not defined, the default *level* is 0.

E.1.3 Projects

The Sentry program works with projects, which can be created, retrieved from disk, saved to disk, and executed.

To create a new project, select the **New** menu item from the **Project** menu.

To retrieve a previously-saved project, select the **Open...** menu item.

To store the current project on disk, select either the **Save** or **Save As...** menu item. If the project already has a name attached to it (i.e., it was previously saved), then you can simply update the project on disk with the **Save** command. If you want to store the project under a different name, select the **Save As...** menu item. A file requester will appear for you to enter a name for the project.

The name of the current project being modified is displayed in Sentry's title bar. It will say "**Project (NONE)**" if no project is currently being modified.

To execute the current project select the **Accept...** menu item.

To continue with execution of a currently running project (paused if the **Control Panel...** button was pressed from the status panel), select the **Continue...** menu item.

E.1.4 Directory Notification

The main Sentry control panel is where you define the directory to examine, the list of commands to invoke on every new and modified file in that directory, and what to do with these files once they have been modified or processed.

The **Directory...** button and input field allow you to specify the specific directory (drawer) to watch.

The **Include Pattern:** and **Exclude Pattern:** input fields allow you to specify AmigaDOS-style filename patterns that will filter the included and excluded files. If you are interested in any file that is modified in the currently-specified directory, specify the pattern “#?” in the **Include Pattern:** field and nothing in the **Exclude Pattern:** field.

You can process existing files within a directory by selecting the **Consider Existing Files?** checkbox. This may come in handy when you want to ensure that all of the files, both old and new, within a directory are processed.

The **Scan Mode:** cycle gadget allows you to specify how often to examine the selected directory. If set to **Continuous**, then all new and current files that have been modified will be processed. If set to **One Time**, only the current files will be processed.

A very handy feature is the **Delete Processed Files?** checkbox. When checked, processed files (those that have been run through the Command List) will be deleted. The **Confirm Before Deleting?** checkbox determines if the processed file is deleted with warning or not.

E.1.5 Command List

Select the **Add...** button to append a new command to the Command List. The Command Specification control panel (shown in Figure E.2) will appear for you to define a new command. This panel will be explained later in this section.

The **Delete** button removes the currently selected command from the List.

The **Move Up** and **Move Down** buttons insert the currently selected command one position above (**Move Up**) or below (**Move Down**) its current



Figure E.2: Sentry's Command Specification control panel.

Environment Variable	Equivalent
SENTRYFULLPATH	Full pathname of the current file.
SENTRYPATH	File's path part only.
SENTRYFILE	File's file part only.
SENTRYFRAME	Current frame number.
SENTRYTOTALFRAMES	Total frames defined.

Table E.1: Sentry's Environment variables.

position in the List.

Command Specification control panel

The **Type**: cycle gadget specifies the class of command that is being defined. If set to **Rexx**, this command can be an ARexx script—even a FRED Script (used in FRED's Invoke ADPro list)—to invoke. If set to **CLI**, this command can be an executable program.

The **Command...** button and input field allow you to specify the actual command (script or CLI program).

The **Arguments**: input field allows you to specify command-specific parameters. For **Rexx** commands, these will be passed to the script, and can be parsed by the script using the **PARSE ARG** or **ARG** commands. For **CLI** commands, these will be passed as CLI arguments.

NOTE To properly execute CLI scripts, you must place the **Execute** command in the **Command...** input field and the name of the CLI script in the **Arguments**: input field.

In some instances you might want to pass the current filename or frame number as an argument. Sentry allows you to use the “environment” or symbolic variables listed in Table E.1 in the **Arguments**: specification.

These variables must be preceeded by a dollar sign (i.e., **\$SENTRYPATH**). So, for example, to send the current filename and frame number (separated by a single space character), you would enter the following:

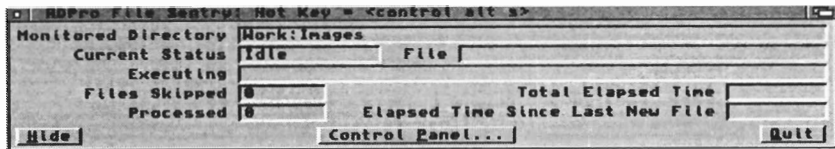


Figure E.3: Sentry's status panel.

\$SENTRYFRAME \$SENTRYFULLPATH

Note that frame numbers begin at 1.

NOTE To properly execute FRED ARexx scripts (i.e., FRED Scripts), you must enter **\$SENTRYFRAME \$SENTRYFULLPATH** in the **Arguments:** field.

Press the **OK** button to preserve the changes made to the command, or **Cancel** to ignore them.

E.1.6 Activation and Monitoring

To activate directory notification on the directory you specified in the **Directory...** input field, select either **Accept...** or **Continue...** from the **Project** menu.

The Sentry status panel (shown in Figure E.3) will appear, showing the progress of the monitoring operation.

Monitored Directory: displays the directory you selected.

Current Status: displays the status of the current file being processed.

File: displays the filename being processed.

Executing: displays the Rexx or CLI command currently being performed on the file.

Files Skipped: displays a counter of files that have not been processed as a result of the **Include Pattern:** and **Exclude Pattern:** filename filters.

Inversely, **Processed:** displays a counter of the files that have been processed.

Total Elapsed Time: displays the duration (in *hours:minutes:seconds* format) since monitoring began.

Elapsed Time Since Last New File:, as its name implies, displays the

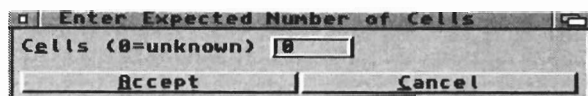


Figure E.4: Sentry's cell information control panel.

duration since the last new file was processed.

The **Hide** button allows you to close the status window but still keep monitoring the directory. This is useful for when you want to unclutter the Workbench screen or, simply, get it "out-of-sight". While hidden, you can invoke the keyboard hotkey to redisplay the panel or select the **Show** or **Show Interface** control from the commodities Exchange program.

The **Control Panel...** button redisplay the main control panel, where you specified the monitoring specifics. The actual monitoring is suspended. You can continue with monitoring by selecting the **Continue...** menu item.

The **Quit** button aborts the directory monitoring and exits the program.

If you have specified a FRED Invoke ADPro script (FRED Script) as one of the commands to be executed, Sentry will recognize that a cell count must be specified. In these instances, the cell information control panel (shown in Figure E.4) will appear.

Sentry uses the information you have defined in this panel to inform any FRED pre-scripts of the total expected number of frames. Although Sentry asks for the number of cells, you can also think of it as asking for the number of frames, since each file that gets processed is considered a cell of length 1.

What is sent to the pre-scripts for cell and frame count is:

```
NumberOfCells = value_user_entered
NumberOfFrames = 0
```

These values are parsed by all pre-scripts. Examine any of the FRED pre-scripts to see how this information is used.

E.1.7 Commodity Support

When you **Accept...** or **Continue...**, Sentry opens a Commodities broker and installs itself as a commodity. Only during processing mode does this happen, as there is no logical reason to be a commodity when it is not doing anything. From the Exchange program, **Enable/Disable**, **Show/Hide**, and **Kill** all behave as expected. The default hotkey is "control alt s" (Ctrl +

Alt + s), but this can be changed from the Sentry icon's Tool Type.

E.1.8 Network and File System Issues

Currently, notification fails on network filesystems, and there is a problem in the **RAM:** filesystem which renders notification useless.

If notification fails or if the monitored directory is in **RAM:**, notification is “emulated” with a timed directory check and possibly a scan. This works with **RAM:**, but currently there is a problem in Commodore's current NFS (Network File System) which causes directory dates to change when a file is created but not closed, rendering this utility almost useless.

If you need to process files on a network, to generate images on one machine and print them out to a color or video printer on another machine, do the following:

- Set up a copy of Sentry on the machine generating the images—generate the images on the machine's local drive.
- Tell Sentry to look for new files on that drive and set up CLI / Shell commands or scripts which copy the files to the local drive of another machine on the network.
- Set up another copy of Sentry on the other network machine, watching for new files on that machine's local drive (they will be sent there by the other machine over the network).
- Instruct *that* copy of Sentry to do what you want with the files on that machine.

Another NFS problem is the ability to allow one program to get a read lock on a file while it is still being written by another program. If this happens, there is not much you can do.

So, this utility will not work if:

- Directory notification does not work on a specific filesystem (i.e., you should get notified when a file is **closed** in a directory on that filesystem). The standard hard drive filesystem works fine.
- The filesystem does not update the directory datestamp when a file is **closed**. NFS falls under this category.
- The filesystem allows read-locks on an exclusive write-locked file. NFS falls under this category as well.
- A program opens, processes, and closes the same file multiple times.

You will get best results with a filesystem which supports true directory notification.

Computer System	Name of Installed Programs	
	Splitz	Joinz
Amiga	<code>splitz</code>	<code>joinz</code>
Macintosh	<code>Split & Join</code>	
MS-DOS	<code>splitz.exe</code>	<code>joinz.exe</code>
Windows	<code>wsplitz</code>	<code>wjoinz</code>

E.1.9 ARexx Issues

Since ARexx cannot properly handle strings with embedded spaces in them, trying to specify the RAM drive through the file requester (which gives the name "Ram Disk:" to it) will cause the program to fail.

Two work-arounds are to:

1. Manually type in **RAM:** when specifying the RAM disk.
2. Relabel the RAM disk using the AmigaDOS **Relabel** command. An example is:

```
Relabel "Ram Disk:" "RamDisk"
```

Note that the new name you want to give the drive does not have the trailing colon. Please consult your AmigaDOS manual for further information.

E.2 Splitz & Joinz

The floppy disk remains the most effect means of transferring data from an Amiga to another type of computer, unless, of course, the data you wish to transfer is too large to fit on a single floppy. In this case, you're out of luck since there is no single program that exists which is available on multiple platforms to split files apart on one computer (like an Amiga), and join them together on another (like an IBM).

ASDG, in response to your needs, has written such a utility and we've even gone to the trouble of supplying you with Macintosh, Windows, and MS-DOS versions, in addition to an Amiga version. The Macintosh version even lets you specify creator and file type!

NOTE You may use these tools on computers which you own or work upon daily. You can provide these tools to others (such as your service

bureau) **only after we receive written permission from your service bureau on their own letterhead stating that they would like to use your copy of Splitz & Joinz.**

Please have your service bureau include the following information in the letter:

- Your name.
- Your ADPro serial number.
- Which version of Splitz & Joinz you are letting them use.

E.2.1 Amiga Set-Up and Usage

To install the Amiga version, use the installation program to extract the Amiga files into the desired disk or directory.

Splitting Files

To split an Amiga file through the Workbench, follow the steps below:

1. Double-click on the **Split** icon.
2. When it asks for the maximum chunk size, enter the number of bytes that each chunk should be. You would like this number to be a few thousand bytes less than the maximum capacity of a disk; it defaults to an PC/MS-DOS 720K 3.5" disk. Select the **Ok** button to proceed.

NOTE If you are transferring files between platforms, you should be using disks that can be read from the "destination" platform. For example, if you will be transferring these files over to the Macintosh, be sure you have the ability to either read Macintosh or PC/MS-DOS disks (which cannot be done normally from your Amiga without the appropriate hardware or software).

3. From the file requester that appears, select the file that you want to split. Either double-click on the filename or single click and press the **Ok!** button.
4. Another file requester will appear asking for the destination directory (entered in the **Drawer** field) to place these chunks, as well as the base name (entered in the **File** field) to use when building the chunk filenames. For example, if the destination were **RAM:** and base name were **psfile**, each of the generated chunk files will be named: **RAM:psfile.001**, **RAM:psfile.002**, and so on.

5. You will be prompted to press the **Return** key before the program writes out each chunk file. You can press the sequence **A + Return** to abort the program.
6. Once all the chunks required to split the original file have been written out, you will be prompted one last time to press the **Return** key to exit the session.

The Amiga version of Splitz can also be invoked through the CLI/Shell. Its command line template is:

```
Splitz filename destination chunksize [PAUSE]
```

You can type **Splitz ?** to display this template.

The optional **PAUSE** keyword (note that you do not include the brackets) is used when you want the program to pause between writing each chunk out to disk.

Joining Files

To join a Splitz file (previously split on the Amiga or other platform) from the Workbench, follow the steps below:

1. Double-click on the **Joinz** icon.
2. A file requester will appear asking for the file that you want to join. You should select the first file in the series of chunks. Note that the file requester is only showing those files (see the pattern in the **Show** field) that end in **.001**.
3. Another file requester will appear asking for the destination filename. The program will use to join all the chunk together into one file using this name. While it is reading each chunk, replace any disk with the next disk in the sequence, if prompted.
4. Once all chunks have been read, you should press the **Return** key to exit the program.

Your joined file should now be in the directory and under the name you specified.

The Amiga version of Joinz can also be invoked through the CLI/Shell. Its command line template is:

```
Joinz basename outputname
```

You can type **Joinz** to display this template.

E.2.2 Macintosh Set-Up and Usage

The Macintosh version of Splitz & Joinz is compatible with System 6 and 7.

To install the Macintosh version, follow the steps below:

1. Use the installation program to extract the Macintosh files into a temporary directory, such as **RAM:.** These files are called **macsplit.hqx** and **macreadme.hqx**.
2. Copy these files onto a PC/MS-DOS formatted floppy disk (you will need a program, such as Consultron's CrossDOS, to do this). For these procedures, we will use only the HQX files.
3. On your Macintosh, run the Apple File Exchange (AFE) program **before** installing the PC/MS-DOS formatted disk into your floppy drive.
4. Select the HQX files in the AFE program and select a destination for them.
5. Hit the **Translate** button. After the files are physically copied, quit the AFE program.
6. Bring up StuffIt (or another utility program that can decode HQX files).
7. Choose **Decode BinHex File** from the **Other** menu item (under the **BinHex** menu).
8. Select the HQX file to decode, and select a place to save Macintosh executable.
9. Finally, hit the **Save** button. Exit StuffIt when done.

Splitz & Joinz should now be ready to run. Note that the Macintosh version is actually one file, and not two as with the other implementations.

Splitting Files

To split a Macintosh file, follow the steps below:

1. Double-click on the **Splitz & Joinz** icon.
2. Select **Split...** from the **File** menu.
3. The program will ask you for the file to split. Select the file and press the **Open** button.
4. The program will then ask you for the name of the first chunk (in the **Save part 1 as:** input field). Also enter the size of each chunk in the **Segment Size:** field. Enter a value for both of these and press the **Save** button.

For each chunk to be saved you will have to accept the operation by pressing the **Save** button.

After the file has been split up into its chunks, you can archive them for later retrieval or transfer them to another platform. Note that you might need to use a utility, such as Apple File Exchange (AFE), to transfer from one platform to another.

Joining Files

To join a Splitz file (previously split on the Macintosh or other platform), follow the steps below:

1. Double-click on the **Splitz&Joinz** icon.
2. Select **Join...** from the **File** menu.
3. The program will ask for the first chunk in the series (*chunkname.001*). Select it and press the **Open** button.
4. The program will then ask for the name of the “restored” file (in the **Save the joined file as:** input field). Enter a new name and press the **Save** button.

For each successive chunk you will have to press the **Open** button.

After all the chunk have be read in, you should have the original file available to you on the Macintosh.

E.2.3 PC/MS-DOS Set-Up and Usage

To install the MS/DOS splitz.exe and joinz.exe, simply copy them onto an MS/DOS floppy disk, and then from the floppy disk onto you MS/DOS systems hard disk.

Splitting Files

To split a PC/MS-DOS file, type the following command at the prompt of the directory where the **SPLITZ.EXE** program is located:

```
SPLITZ.EXE filename destination chunksize /P
```

substituting each parameter with the required value.

You can type **SPLITZ.EXE ?** to display this template.

The optional **/P** keyword (note that you do not include the brackets) is used when you want the program to pause between writing each chunk out to disk.

Joining Files

To join a Splitz file (previously split on a PC/MS-DOS machine or other platform), type the following command at the prompt of the directory where the JOINZ.EXE is located):

JOINZ.EXE *basename outputname*

substituting each parameter with the required value.

You can type JOINZ.EXE to display this template.

E.2.4 Windows Set-Up and Usage

To install the WINDOWS wsplitz.exe and wjoinz.exe, simply copy them onto an MS/DOS floppy disk, and then from the floppy disk into windows directory on your hard disk (usually C:\windows).

1. From the Program Manager, select the program group in which you would like Splitz & Joinz to reside.
2. Select the **New** menu item from the Program Manager's **File** menu.
3. Select **Program Item** and click on the **OK** button.
4. In the window that appears, enter **Splitz** for the description, press the **Tab** key, then enter C:\WINDOWS\WSPLITZ.EXE.

The WSPLITZ icon should now appear in the selected program group. Repeat steps 2, 3 and 4 for WJOINZ.

Splitting Files

To split a Windows file, follow the steps below:

1. Double-click on the **WSplitz** icon to run it.
2. Select the file to split from the directory list on the left side of the panel.
3. Select the size (in number of bytes) of each chunk in the **Chunk size:** input field.
4. Select the destination and output base name for the chunks in the **Output File Base Name** input field.
5. Press the **Accept** button to initiate the split operation.

Once all the chunks have been created, you can store them for later retrieval or transfer them to another platform for joining back together.

Joining Files

To join a Splitz file (previously split under Windows or another platform), follow the steps below:

1. Double-click on the **WJoinz** icon to run it.
2. Select the first chunk by finding it in the left-side directory listing.
3. In the **Output File:** input field, enter the name that the joined file will take.
4. Press the **Accept** button to initiate the operation.

E.3 ZapDPI

Summary Professional Page (ProPage) 2.0 crashes on IFF files saved by ADPro 2. This is a short term work-around (ProPage 3.0 and later does not have this problem).

As of ADPro 2.0.0, ASDG began using an IFF chunk called “DPI” to store resolution information. This chunk is an endorsed part of the IFF standard and was part of the IFF definition published by Commodore at the 1991 Developer’s Conference.

Unfortunately, Gold Disk Incorporated, makers of Professional Page, had been using their own proprietary chunk (also called **DPI**) which they did not register with Commodore. Since the chunks are named the same, Professional Page will attempt to use the **DPI** information ADPro writes in its IFF files and, as a result, will crash (it tries to divide by zero).

If you are still using a version of ProPage that has behaves this way, you can use the ZapDPI utility to remove and restore our **DPI** chunks from IFF files written by ADPro version 2.

From the ADPro main window (if you own ARexx) you can run the **ZapDPI.adpro** ARexx script to remove **DPI** chunks. Or, you can run the **ReDPI.adpro** script to put **DPI** chunks back into IFF files saved by ADPro 2. Finally, you can run **CheckDPI.adpro** to find out if a given file has a **DPI** chunk or not.

From the CLI/Shell, you can remove **DPI** chunks using the **-dpi** command line option. To put **DPI** chunks back in, you would use the **-DPI** command line option. To check a file, use the **-c** command line option.

From the Workbench, ZapDPI will let you inspect each selected file and act on each one individually.

Command Line Option	Function
-dpi	Remove DPI chunk
-DPI	Insert DPI chunk
-c	Check for DPI chunk

E.4 ReleaseNotes

The **ReleaseNotes** utility is used to display the release notes for ADPro from within the program. It should be available in the **ADPRO:** directory.

The **ReleaseNotes** icon contains the following tool types:

FILE=*filename*

If present, this file will be displayed in the file requester when the program is launched without a filename. Notes files typically have a **".notes"** extension.

VIEWER=*utility*

If present, this utility will be used to display the selected notes file. When run from the command line (Shell), this utility will be used to display the specified file.

VIEWER_OPTS=*commands*

If present, these options will be sent to the selected viewer. Use this tool type to specify any commands above and beyond a public screen argument.

PUBSCREEN=*public_screen_name*

If present, the viewer will be attempted to open on the specified public screen. If not present or if the named public screen does not exist, then the file viewer will open on the default public screen.

If you double-click on the **ReleaseNotes**, then the settings defined in the icon will be used. If not file was specified, then a file requester will appear. If no viewer was specified, then the **More** utility will be used for AmigaOS 2.04 and 2.1 or **MultiView** for AmigaOS 3.0 and later.

Index

A

- A-HAM, 109, 151
- A-RES, 109, 151
- A2410, 127, 163
- About, 7
 - command, 283, 350
- About ADPro, 87
- ABS_SCALE, 446
- activating ADPro, 93, 342
- activating ADPro's window
 - via AppIcon, 93
 - via hot-key, 130
- ADP_FRED: directory, 282
- ADPRO: directory, 86, 484
- ADPro.prefs, 341, 344
- ADPRO_DISPLAY, 456
- ADPRO_EXIT, 351
- ADPRO_TO_BACK, 352
- ADPRO_TO_FRONT, 352,
 - 365–369, 372, 373, 456
- ADPRO_UNDISPLAY, 456
- ADProScripts, 319
- AGA, 36, 108, 290
- ALPHA, 132
- alpha channel, 483
 - embedded, 99
- alpha channel buffer, 36
- alpha compositing, 99
- ALPHA_ONLY, 378
- ALPHAMEM, 378
- Amiga, 273
 - module name, 463
- AmigaOS
 - 2.04, 11, 30, 106, 256, 512
 - 2.1, 11, 30, 105, 512
 - 3.0, 11, 30, 512
- AmigaOS 2.04, 273
- AmigaOS 2.1, 273
- AmigaOS 3.0, 273
- ANIM, 134, 166, 400
- AnimOps, 296
 - Cinemorph, 297
 - Compositor, 300
 - Time.Stretch, 304
- anti-alias, 484
- Antique, 219
 - ARexx control, 428
- AppIcon, 93
 - activating ADPro, 93
 - changing the icon, 130
- Apply_Map, 184, 185, 219, 220,
 - 483
 - ARexx control, 428
- AppMenu, 93
 - activating ADPro, 93
- APPSCRIPT, 93, 341, 343
- AppWindow, 93
 - startup script, 341, 343
- ARexx
 - addressing ADPro, 309
 - creating scripts, 312
 - general rules, 312
 - invoking commands, 311
 - keyboard macros, 313
 - launching programs, 313
 - port name, 309
 - pre-written scripts, 314
 - RC, 310–312
 - result variable, 310–312
 - results, 310
 - selecting scripts, 316

suggested contents, 312

ARexx commands

- ABS_SCALE, 446
- ADPRO_DISPLAY, 456
- ADPRO_EXIT, 351
- ADPRO_TO_BACK, 352
- ADPRO_TO_FRONT, 352
- ADPRO_UNDISPLAY, 456
- AVAIL_MODES_ONLY, 457
- BLUE, 354
- BRIGHTNESS, 355
- CLEAR_RAW, 457
- CLEAR_RENDERED, 457
- CLOSE_RENDER_SCREEN, 458
- CONTRAST, 355
- CURRENT_MEM_SIZE, 350
- DISPLAYMESSAGE, 372
- DITHER, 460
- DITHER_AMOUNT, 461
- EXECUTE, 455
- GAMMA, 356
- GET_SCREEN_MODE, 370
- GETDIR, 369
- GETFILE, 367
- GETFILES, 368
- GETLIST, 371
- GETNUMBER, 366
- GETSTRING, 365
- GREEN, 354
- IMAGE_TYPE, 461
- INTERFACE, 353
- INTERFACE_SIZE, 353
- LAST_LOADED_IMAGE, 364
- LAST_SAVED_IMAGE, 364
- LFORMAT, 375
- LISTVIEW, 371
- LOAD, 375
- LOAD_DEFAULTS, 373
- LOAD_GUI, 376
- LOAD_TYPE, 380
- LOADER, 376
- miscellaneous, 364
- MPPOKE, 361
- OBTAIN_ADPRO, 374
- OFORMAT, 427
- OKAY1, 372
- OKAY2, 373
- OKAYN, 373
- OPERATE, 427
- OPERATE_GUI, 428
- OPERATOR, 427
- ORIENTATION, 379
- PAUSE, 365
- PCONTRAST, 359
- PCT_SCALE, 446
- PGETWB, 362
- PIXEL_ASPECT, 462
- PIXEL_RESOLUTION, 462
- PIXELPEEK, 361
- PIXELPOKE, 361
- PLOAD, 362
- POFFSET, 358
- PPOKE, 360
- PSAVE, 363
- PSORT, 360
- PSTATUS, 357
- PTOTAL, 357
- PUSED, 358
- PWIDTH, 356
- RDEPTH, 457
- RED, 353
- RELEASE_ADPRO, 374
- RENDER_TYPE, 455
- RMODE, 456
- SAVE, 397
- SAVE_DEFAULTS, 374
- SAVE_GUI, 398
- SAVER, 398
- SCREEN_TO_FRONT, 352
- SCREEN_TYPE, 458
- SERIAL_NUMBER, 350
- SET_ADPRO_MODE, 351
- SET_ADPRO_PUBLIC, 351
- SET_RENDER_MODE, 459
- SETPALINFO, 359
- SFORMAT, 397
- VERSION, 350
- XSIZE, 461
- YSIZE, 461

ARexx loader formats

- ALPHA, 380
 - ANIM, 380
 - BACKDROP, 381
 - BACKLINE, 382
 - BMP, 383
 - CDXL, 383
 - CLIPBOARD, 384
 - DPAINT, 385
 - DPIIE, 385
 - DV21, 386
 - FC24, 386
 - FLC, 386
 - FRACTAL2000, 387
 - FRAMEGRABBER, 387
 - FRAMESTORE, 388
 - GIF, 388
 - HAME, 389
 - ICO, 389
 - ICON, 389
 - IFF, 389
 - IMPULSE, 390
 - IV24, 390
 - JPEG, 391
 - MACPAINT, 391
 - PAR_PEG, 392
 - PCX, 392
 - POINTER, 393
 - QRT, 393
 - SCREEN, 393
 - SCULPT, 394
 - TEMP, 395
 - UNIVERSAL, 395
 - VLAB, 395
 - YUVN, 395
 - ARexx loader options
 - FORCE_PALETTE, 375
 - NOPAD, 39, 384, 389, 392
 - ONLYPAD, 39
 - ARexx saver formats
 - A2410, 398
 - ANIM, 400
 - BMP, 401
 - CDXL, 401
 - CLIPBOARD, 403
 - DPAINT, 403
 - DPIIE, 404
 - FC24, 404
 - FRAMEBUFFER, 406
 - FRAMESTORE, 406
 - GIF, 407
 - HAME, 407
 - HARLEQUIN, 408
 - ICON, 410
 - IFF, 411
 - IMPULSE, 411
 - IV24, 411
 - JPEG, 412
 - OPALVISION, 413
 - PCX, 414
 - PICASSO, 415
 - PREFPRINTER, 416
 - QRT, 419
 - RESOLVER, 419
 - RETINA, 421
 - SCULPT, 422
 - SEPARATE, 422
 - TEMP, 425
 - VT_BUFFER, 425
 - ARexx saver options, 398
 - AS, 93, 343
 - As Alpha, 36
 - ASALPHA, 377
 - ASCII, 202
 - ASL
 - file requester, 25
 - font requester, 28
 - screen mode requester, 29
 - aspect
 - image, 40
 - pixel, 40
 - aspect ratio, 40
 - asynchronous, 92
 - auto-scroll, 274
 - AVAIL_MODES_ONLY, 457
 - Available Modes Only, 108, 275, 278
- ## B
-
- BACKDROP, 135
 - BACKLINE, 137
 - Backward Brick, 202

- Backward Diagonal, 202
- balancing, 114, 115, 480
 - ARexx control, 353
 - brightness, 116
 - contrast, 116
 - control panel, 118
 - gamma, 117
- BEHIND, 340, 342
- BH, 342
- BLUE, 354
- Blur, 220
 - ARexx control, 428
- blur, 220
- BMP, 138, 168
- BRIGHTNESS, 355
- brightness, 116, 219, 480
- Broadcast_Limit, 221
 - ARexx control, 429
- buffer
 - alpha channel, 36
 - raw, 36
 - rendered, 36
 - temporary, 36
- button, 21
- Button Interface, 90, 129
 - invoking loaders, 94
 - invoking operators, 104
 - invoking savers, 103
 - setting compose mode, 95

C

- CDXL, 139, 169, 401
- check box, 23
- chooser, 30
- Cinemorph, 297
- CLEAR_RAW, 457
- CLEAR_RENDERED, 457
- CLIPBOARD, 128, 138, 171
- Clipboard, 287
 - loading from, 128
 - saving to, 128
- CLOSE_RENDERER_SCREEN, 458
- CLUT, 184
- CLUT chunk, 483
- CMAP, 213, 214

- cmy separations, 211
- cmyk separations, 212
- Collapse, 222
 - ARexx control, 430
- color controls, 39, 113, 114, 219, 428
- color look-up table, 119, 220
- color map, 34, 35, 37, 114–118
 - linear, 114
- color mapped, 108
- color mode, 30
- color picking, 122, 480
- color-mapped modes, 113
- Color_To_Gray, 224, 225
 - ARexx control, 431
- Colorize, 225
 - ARexx control, 432
- Colors, 30, 107, 122
- colors
 - available, 274
 - limiting selection, 108
 - offset color zero, 124
 - total, 123
 - used, 123
- Colors (Used), 279
- command line interface, 9, 86, 282, 498
- commodity
 - hot-key, 342
 - hot-key activation, 130
- COMPMIX, 377
- COMPOFFSET, 377
- Compose, 95
- Compositing, 95
- compositing, 95
 - mix level, 98
 - restrictions, 96
 - screen, 95
 - transparent color, 98
 - transparent color range, 98
- compositing an image, 99
- compositing options
 - ALPHA_ONLY, 378
 - ALPHAMEM, 378
 - ASALPHA, 377
 - COMPMIX, 377

- COMPOFFSET, 377
- COMPTRANS, 377
- COMPTRANSTO, 377
- NOPAD, 378
- ONLYPAD, 378
- USE_ALPHA, 378
- Compositor, 300
 - alpha channel, 301
 - ARexx hooks, 302, 303
 - components, 301
 - mix level, 302
 - positioning, 302
 - projects, 300
 - results, 303
 - sequences, 301
 - transparency, 302
- COMPTRANS, 377
- COMPTRANSTO, 377
- continuous, 198
- CONTRAST, 355
- contrast, 116, 118, 478, 480
- control panel, 21
- control screen, 20
- controls, 88
- Convolve, 227
 - ARexx control, 433
- Crop_Image, 228
 - ARexx control, 433
- Crop_Visual, 32, 230, 480
 - ARexx control, 434
- CURRENT_MEM_SIZE, 350
- CUST, 455
- Custom Palette, 122, 126, 279, 482
- custom palette, 122
- customizing ADPro, 129
- CX_POPKEY, 130, 342
- cycle gadget, 22

D

- data flow, 35, 114
- data type
 - conversion during load, 35
 - display, 35

- raw, 34, 35, 38, 39, 103, 107, 111, 114, 118, 125, 126, 219, 253
- rendered, 34, 35, 39, 103, 107
- requirements, 35
- data types
 - internal, 34
- DCTV, 151, 231
 - ARexx control, 434
- default notes file, 512
- DEFAULTFILE, 341, 343
- defaults file
 - custom, 341, 343
 - not loading it, 341, 344
 - not saving it, 341, 344
- Define_Pxl_Aspect, 204, 231
 - ARexx control, 434
- DeInterlace, 232
 - ARexx control, 436
- depth
 - locked, 102
- depth of separation, 213
- developers, 489
- DF, 343
- DFLT, 459
- Digi-View 3.0, 141
- Displace_Pixel, 233
 - ARexx control, 436
- display box, 24
- display module, 105
- displaying an image, 113, 274
- DISPLAYMESSAGE, 372
- displays
 - Amiga, 273, 463
 - EGS, 275, 463
 - FC24, 276, 463
 - OpalVision, 277, 463
 - Picasso, 277, 463
 - Retina, 278, 463
 - Window, 279, 464
- DITHER, 460
- dither, 126, 224, 476
 - ARexx control, 460
 - types, 109
 - used in VUI, 32
- dither amount

ARexx control, 461
DITHER_AMOUNT, 461
DPAINT, 14, 140, 171
DPI, 33
DPIIE, 141, 173
drag and drop, 93
dropping icons, 93
DV21, 141
Dynamic HAM, 109, 151
Dynamic Hi-Res, 109, 151
Dynamic_Range, 234
 ARexx control, 436

E

edit palette, 119
EGS, 275
 module name, 463
EHB, 108, 113, 151, 163, 179, 189,
 190, 208, 357, 358, 455
embedded alpha channel, 99
EN, 343
Encapsulated PostScript, 192
ENHANCED, 341, 343
enhanced palette, 13, 122, 127,
 360
 ARexx control, 356
 control, 341, 343
EPS, 192
EXECUTE, 455
Execute, 88, 104, 111, 113, 120,
 125, 274
 ARexx control, 455
 command, 35, 482
Execute ARexx Script, 316
Exit
 ARexx control, 351
exiting ADPro, 87
exiting FRED, 283
external alpha channel, 99

F

fan-folded, 198
FanFold, 198
FARGO Primera, 198

FASTMEMONLY, 341, 343
FC24, 142, 173, 276, 478
 module name, 463
file requester, 24
 filtering files, 26
 patterns, 26
FLC, 142, 386
floating, 92
floating list
 invoking loaders, 94
 invoking operators, 104
 invoking savers, 103
floating lists, 92
Floyd, 202
focus indicator, 88
font
 on custom screens, 129
 on Workbench, 129
font requester, 28
FONTS: directory, 28
form feed, 198
Forward Brick, 202
Forward Diagonal, 202
FRACTAL2000, 143
FRAMEBUFFER, 177
FRAMEGRABBER, 147, 478
FRAMESTORE, 149, 177
FRED, 281
 AnimOps, 297
 cell arrangement, 287
 cell duration, 289
 cell information, 288
 cell selection, 286
 cells, 283, 286
 Clipboard, 287
 frames, 283, 286
 Invoke ADPro, 292, 293
 linking with ADPro, 291
 main process script, 295
 post processing, 293
 post-process script, 296
 pre processing, 293
 pre-process script, 294
 Process, 292
 reference, 465
 sequences, 283

stamps, 290
fred.post script, 296
FREDScripts, 323

G

gadget, 21
GAMMA, 356
gamma, 117, 219, 480
GCR, 213
genlock, 123
GET_SCREEN_MODE, 351, 370,
459, 463
GETDIR, 369
GETFILE, 367
GETFILES, 368
GETLIST, 371
GETNUMBER, 366
GETSTRING, 365
getting acquainted, 4
GIF, 149, 179
Gray_To_Color, 96, 234, 481
ARexx control, 437
grayscale
color mode locked, 108
GREEN, 354

H

Halftone A, 202
Halftone B, 202
Halve, 235
ARexx control, 437
HAM, 108, 111, 113, 151, 163,
179, 189, 190, 208, 274,
357, 358, 455
HAM8, 108, 113, 180, 189, 357,
358, 455
HAME, 149, 179, 407
HARLEQUIN, 181
Hi color, 108
highlight colors, 121
Hist.Equalization, 235
ARexx control, 438
Horizontal, 202
horizontal

aspect ratio, 32
Horizontal.Flip, 95, 235
ARexx control, 438
hot-key, 130
hot-key sequence, 342
HSI support, 153
HSV, 55, 120, 122

I

ICO, 150, 389
ICON, 150, 183, 389, 410
IFF, 151, 184, 231, 411
internal pixel aspect, 253
image aspect, 40
image buffer, 36
Image Compositing, 477
control screen, 95
Image Information, 35, 88, 90
image masking, 102, 132, 476
image operators, 39
IMAGE_TYPE, 461
IMPULSE, 152, 184
ink correction, 214
input field, 22
input focus, 88
installation, 14
Installer, 14
Intensity.Range, 236
ARexx control, 438
INTERFACE, 353
interface
list, 340, 343
starting up minimized, 340,
343
INTERFACE_SIZE, 353
Interlace, 237
ARexx control, 438
IV24, 152, 185

J

JPEG, 153, 187

K

- keyboard equivalents, 24
- keyboard macros, 313
- keyboard qualifiers
 - FRED, 465
- keyboard shortcuts, 344
- KillTemp, 237
 - ARexx control, 439

L

- Landscape, 95
- last-minute changes, 87
- LAST_LOADED_IMAGE, 364
- LAST_SAVED_IMAGE, 364
- launching a script at startup, 342, 344
- LFORMAT, 375
- LI, 343
- Line_Art, 238
 - ARexx control, 439
- list
 - scrolling contents, 88, 92
- List Interface, 88, 129
 - invoking loaders, 94
 - invoking operators, 104
 - invoking savers, 103
- LISTINTERFACE, 340, 343
- LISTVIEW, 371
- LOAD, 375
- LOAD_DEFAULTS, 373
- LOAD_GUI, 376
- LOAD_TYPE, 380
- LOADER, 376
- loader
 - internal work, 38
- loaders, 37, 94
 - ALPHA, 132
 - ANIM, 134
 - BACKDROP, 135
 - BACKLINE, 137
 - BMP, 138
 - CDXL, 139
 - CLIPBOARD, 138
 - DPAINT, 140
 - DPIIE, 141
 - DV21, 141
 - FC24, 142
 - FLC, 142
 - FRACTAL2000, 143
 - FRAMEGRABBER, 147
 - FRAMESTORE, 149
 - GIF, 149
 - HAME, 149
 - ICO, 150
 - ICON, 150
 - IFF, 151
 - IMPULSE, 152
 - invoking, 94
 - invoking from Button Interface, 94
 - invoking from floating lists, 94
 - invoking from List Interface, 94
 - IV24, 152
 - JPEG, 153
 - MACPAINT, 154
 - PAR_PEG, 154
 - PCX, 155
 - POINTER, 155
 - pseudo modules, 315
 - QRT, 156
 - SCREEN, 157
 - SCULPT, 158
 - TEMP, 159
 - UNIVERSAL, 94, 131
 - VLAB, 160
 - YUVN, 160
- Loaders2 directory, 134, 380
- loading, 94
- loading cell image, 287

M

- MACPAINT, 154
- magenta correction, 214
- MAXMEM, 36, 240, 340, 343
- Median_Filter, 238
 - ARexx control, 439
- memory
 - using fast, 341, 343
- memory requirements

- contiguous, 11
- for color images, 12
- for grayscale images, 12
- limiting, 13, 240, 340, 343
- reducing, 13
- menu bar, 20
- menu descriptions, 346, 466
- meter window, 31
- MicroIllusions, 266
- MINIMIZED, 340, 343
- MM, 240, 343
- Mode, 30
- mode
 - unnamed, 274
- module
 - indicator, 94
 - launching, 90, 92
- modules, 37
- Mosaic, 239
 - ARexx control, 440
- motion blur, 486
- MPPOKE, 361

N

- NE, 343
- Negative, 240
 - ARexx control, 440
- NL, 344
- NOENHANCED, 341, 343
- NOLOADDEFAULTS, 341, 344
- NOLOADSETTINGS, 344
- NOPAD, 39, 378
- NOSAVEDEFAULTS, 341, 344
- NOSAVESETTINGS, 344
- notes viewer, 512
 - opening on public screen, 512
 - options, 512
- NS, 344
- NTSC, 107, 224
 - aspect ratio, 252
 - pixels, 252
 - playback speed, 291
- nudge, 97
- number of colors, 123, 357

O

- OBTAIN_ADPRO, 374
- Offset Color Zero, 124, 279
- OFORMAT, 427
- OKAY1, 372
- OKAY2, 373
- OKAYN, 373
- ONLYPAD, 378
- ONLYPADD, 39
- OpalPaint, 240
 - ARexx control, 441
- OPALVISION, 188
- OpalVision, 277
 - module name, 463
- OPERATE, 427
- OPERATE_GUI, 428
- operating, 104
- OPERATOR, 427
- operators, 39, 104
 - Antique, 219, 428
 - Apply_Map, 219, 220, 428, 483
 - Blur, 220, 428
 - Broadcast_Limit, 221, 429
 - Collapse, 222, 430
 - Color_To_Gray, 224, 225, 431
 - Colorize, 225, 432
 - Convolve, 227, 433
 - Crop_Image, 228, 433
 - Crop_Visual, 32, 230, 434
 - DCTV, 231, 434
 - Define_Pxl_Aspect, 231, 434
 - DeInterlace, 232, 436
 - Displace_Pixel, 233, 436
 - Dynamic_Range, 234, 436
 - Gray_To_Color, 96, 234, 437, 481
 - Halve, 235, 437
 - Hist_Equalization, 235, 438
 - Horizontal_Flip, 95, 235, 438
 - Intensity_Range, 236, 438
 - Interlace, 237, 438
 - invoking, 104
 - invoking from Button
 - Interface, 104

- invoking from floating lists, 104
 - invoking from List Interface, 104
 - KillTemp, 237, 439
 - Line_Art, 238, 439
 - Median_Filter, 238, 439
 - Mosaic, 239, 440
 - Negative, 240, 440
 - OpalPaint, 240, 441
 - Patter, 441
 - Pattern, 241
 - Polar_Mosaic, 242, 442
 - pseudo modules, 315
 - Rectangle, 243, 443
 - Rectangle_Visual, 244, 444
 - Rem_Isolated_Pxls, 245, 444
 - Rendered_To_Raw, 247, 444
 - Roll, 248, 445
 - Rotate, 95, 235, 248, 269, 445
 - Saturation, 251, 446
 - Scale, 32, 251, 253, 446
 - Sim_Print, 254, 447
 - Text_Visual, 255, 448
 - Tile, 261, 452
 - Tile_Visual, 261, 265, 452
 - TPort_Controller, 266, 452
 - Twirl, 266, 453
 - Vertical_Flip, 95, 269, 454
 - Ordered, 202
 - ORIENTATION, 379
 - orientation, 198
 - during a load, 95
 - overscan, 29
 - ARexx control, 459
- ## P
-
- page gap, 198, 199
 - PAL, 107, 459
 - pixels, 252
 - playback speed, 291
 - palette, 119
 - 256 color editor, 119
 - accuracy, 127
 - colors used, 123
 - contrast, 125
 - contrasting colors, 125
 - custom, 122
 - custom adjustments, 122
 - edit, 119
 - getting the Workbench
 - colors, 126
 - highlight colors, 121
 - loading, 125
 - locked, 37, 126
 - offset of color zero, 124
 - saving, 125
 - sort direction, 124
 - status, 126
 - total colors, 123
 - unlocked, 126
 - use, 122
 - palette accuracy
 - Shell option, 343
 - Tool Type option, 341
 - palette mapped, 108
 - palettes
 - floating tool, 92
 - paper type
 - continuous, 198
 - fan-folded, 198
 - FanFold, 198
 - rolled, 198
 - Single, 198
 - PAR_PEG, 154
 - Pattern, 241
 - ARexx control, 441
 - PAUSE, 365
 - PCONTRAST, 359
 - PCT_SCALE, 446
 - PCX, 155, 189
 - PGETWB, 362
 - PICASSO, 190
 - Picasso, 277
 - module name, 463
 - pixel aspect, 40
 - pixel representation, 114
 - PIXEL_ASPECT, 462
 - PIXEL_RESOLUTION, 462
 - PIXELPEEK, 361
 - PIXELPOKE, 361

- plane of separation, 213
- PLOAD, 362
- POFFSET, 358
- POINTER, 155
- Polar_Mosaic, 242
 - ARexx control, 442
- Portrait, 95
- posters, 198
- POSTSCRIPT, 128, 191
- PostScript
 - angles, 193
 - aspect, 196
 - automatic feed, 197
 - bleed, 196
 - copies, 197
 - crop marks, 196
 - densities, 193
 - image dimensions, 193
 - image offset, 193
 - manual feed, 197
 - negative, 197
 - orientation, 195
 - page dimensions, 193
 - positive, 197
 - registration marks, 195
- PostScript™ printing, 128
- PPOKE, 360
- Preferences
 - paper type, 198
- preferences, 129
- Preferences printing, 128
- PREFPRINTER, 128, 198
 - dither mask size, 201
 - dither type selection, 202
 - DPI, 201
 - effective colors, 201
 - effective gray shades, 201
 - gray ramp, 202
 - mask size selection, 202
- Primera, 128, 198
- printing an image, 127
- PS, 342
- PSAVE, 363
- pseudo modules, 315
- PSORT, 360
- PSTATUS, 357

- PTOTAL, 357
- public screen
 - display in window, 105
 - opening as, 129
 - opening on, 129, 342
 - render window, 279
- PUBSCREEN, 340, 342
- PUSED, 358
- PWIDTH, 356

Q

- QRT, 156, 208

R

- radio button, 23
- raw space, 36
- RC, 310–312
- RD, 313, 314, 344
- RDEPTH, 457
- Rectangle, 243, 477
 - ARexx control, 443
- Rectangle_Visual, 244, 477
 - ARexx control, 444
- RED, 353
- Redisplay, 88, 105, 113, 119
 - command, 38, 456
- regional processing, 483
- registration card, 7
- release notes, 17, 87
- RELEASE_ADPRO, 374
- ReleaseNotes, 512
 - FILE, 512
 - PUBSCREEN, 512
 - VIEWER, 512
 - VIEWER_OPTS, 512
- Rem_Isolated_Pxls, 245, 247
 - ARexx control, 444
- Remove Isolated Pixels, 245
- render depth
 - locked, 102
- render screen, 273
 - visible area, 106
- render space, 36

- alpha channels, 36
- render window, 279
- RENDER_TYPE, 455, 459
- rendered data
 - on high color devices, 113
 - screen mode, 102
- Rendered_To_Raw, 247
 - ARexx control, 444
- Replace, 95
- ReSEP, 213, 214
- resolution, 33, 107, 111, 224, 231, 232, 356
- RESOLVER, 208
- RETINA, 210
- Retina, 278
 - module name, 463
- REXX: directory, 313, 314
- REXXDIR, 313, 314, 342, 344
- RGB, 55, 120, 122
- RGB files, 483
- rgb separations, 211
- RGB8, 152, 185
- RGBN, 152, 185
- RIP, 245, 247
- RMODE, 456
- rocker switch, 22
- Roll, 248
 - ARexx control, 445
- rolled, 198
- Rotate, 95, 235, 248, 269
 - amount of rotation, 250
 - ARexx control, 445
 - blurring, 250
 - center of rotation, 250
 - centering, 250
 - circle radius, 250
 - precision, 250
 - Rotate Circle, 248

S

- Saturation, 251
 - ARexx control, 446
- SAVE, 397
- SAVE_DEFAULTS, 374
- SAVE_GUI, 398

- SAVER, 398
- savers, 39, 103
 - A2410, 163
 - ANIM, 166
 - BMP, 168
 - CDXL, 169
 - CLIPBOARD, 171
 - DPAINT, 171
 - DPIIE, 173
 - FC24, 173
 - FRAMEBUFFER, 177
 - FRAMESTORE, 177
 - GIF, 179
 - HAME, 179
 - HARLEQUIN, 181
 - ICON, 183
 - IFF, 103, 184
 - IMPULSE, 184
 - invoking, 103
 - invoking from Button
 - Interface, 103
 - invoking from floating lists, 103
 - invoking from List Interface, 103
 - IV24, 185
 - JPEG, 187
 - OPALVISION, 188
 - PCX, 189
 - PICASSO, 190
 - POSTSCRIPT, 191
 - PREFPRINTER, 198
 - pseudo modules, 315
 - QRT, 208
 - RESOLVER, 208
 - RETINA, 210
 - SCULPT, 211
 - SEPARATE, 211
 - TEMP, 215
 - VT.BUFFER, 216
- saving, 102
- SCALE, 212
- Scale, 32, 251, 253
 - ARexx control, 446
- SCREEN, 157
- screen

- auto-scroll, 274
 - opening behind others, 340, 342
 - opening on public, 340
 - overscan, 29
 - public, 342
 - virtual, 29
- screen area
 - restrictions, 106
- Screen Controls, 107
- Screen Information, 88, 90
- screen mode, 39
 - locked, 102
- screen mode requester, 28
- screen settings
 - SHALLOW, 465
 - shallow depth, 465
- SCREEN_TO_FRONT, 352
- SCREEN_TYPE, 458, 459
- script
 - launched at startup, 342
- scroller, 24
- SCULPT, 158, 211
- selling ADPro, 7
- Sentry, 497
- SEPARATE, 211
 - cmy, 211
 - cmyk, 212
 - depth, 213
 - plane, 213
 - rgb, 211
 - type, 213
 - using with Professional Page, 214
- sequence
 - adding images, 284
 - creating new, 283
 - loading, 285
 - processing, 296
 - saving, 285
- serial number, 7, 87
 - ARexx control, 350
- SERIAL_NUMBER, 350
- serializing ADPro, 16
- Set ADPro's Font, 129
- Set ADPro's Screen, 129
- Set Render Screen, 88, 105, 273-279, 455
 - colors, 123
- SET_ADPRO_MODE, 351
- SET_ADPRO_PUBLIC, 351
- SET_RENDER_MODE, 370, 459
- SETCPU, 11
- SETPALINFO, 359
- SETTINGS, 85, 341
- settings
 - custom, 341
 - not loading it, 341
 - not saving it, 341
- SFORMAT, 397
- shadows, 477
- SHALLOW, 283, 290, 465
- SHAM, 109, 151
- Shell, 9
 - starting ADPro, 86
 - starting FRED, 282
 - starting Sentry, 498
- Shell options, 342, 465
 - ADPro, 86
 - APPSCRIPT, 343
 - AS, 93, 343
 - BEHIND, 342
 - BH, 342
 - DEFAULTFILE, 343
 - DF, 343
 - EN, 343
 - ENHANCED, 343
 - FASTMEMONLY, 343
 - FRED, 283
 - LI, 343
 - LISTINTERFACE, 343
 - MAXMEM, 343
 - MINIMIZED, 343
 - MM, 343
 - NE, 343
 - NL, 344
 - NOENHANCED, 343
 - NOLOADDEFAULTS, 344
 - NOLOADSETTINGS, 344
 - NOSAVEDEFAULTS, 344
 - NOSAVESETTINGS, 344
 - NS, 344

- obsolete, 344
- PS, 342
- PUBSCREEN, 342
- RD, 344
- REXXDIR, 344
- SHALLOW, 283, 465
- SS, 344
- STARTSCRIPT, 344
- STARTUP, 344
- Sim_Print, 254
 - ARexx control, 447
- Single, 198
- slider, 23
- specifying the REXX: directory,
 - 342, 344
- splitting frames, 148
- Splitz & Joinz, 505
- SS, 344
- STARTSCRIPT, 342, 344
- STARTUP, 344
- SUP72, 459
- system requirements, 11

T

- technical support, 6
- TEMP, 159, 215, 425
- temporary buffer, 35
- text field, 24
- Text_Visual, 255
 - ARexx control, 448
- third-party developers, 489
- Tile, 261
 - ARexx control, 452
- Tile_Visual, 261, 265
 - ARexx control, 452
- time lapse, 486
- Time_Stretch, 304
- Tool Type options, 340, 465
 - APPSCRIPT, 93, 341
 - BEHIND, 340
 - CX_POPKEY, 342
 - DEFAULTFILE, 341
 - ENHANCED, 341
 - FASTMEMONLY, 341
 - FRED, 282

- LISTINTERFACE, 340
- MAXMEM, 240, 340
- MINIMIZED, 340
- MM, 240
- NOENHANCED, 341
- NOLOADDEFAULTS, 341
- NOSAVEDEFAULTS, 341
 - obsolete, 342
- PUBSCREEN, 340
- REXXDIR, 342
- SETTINGS, 341
- SHALLOW, 465
- STARTSCRIPT, 342
- TPort_Controller, 266
 - ARexx control, 452
- Transport Controller, 266
- True color, 108
- Twirl, 266
 - amount of twirl, 268
 - ARexx control, 453
 - blurring, 268
 - center of twirl, 267
 - centering, 267
 - precision, 268
 - Twirl Circle, 267
 - twirl radius, 268
- type of separation, 213

U

- UCR, 213
- UNIVERSAL, 94, 131
 - actual loader used, 94
- unnamed mode, 274
- USE_ALPHA, 378
- User Commands, 315
- user interface
 - button, 21
 - check box, 23
 - chooser, 30
 - control panel, 21
 - control screen, 20
 - cycle gadget, 22
 - definitions, 19
 - descriptions, 19
 - display box, 24

- file requester, 24
- font requester, 28
- gadget, 21
- input field, 22
- keyboard equivalents, 24
- menu bar, 20
- meter window, 31
- radio button, 23
- rocker switch, 22
- screen mode requester, 28
- scroller, 24
- slider, 23
- text field, 24

V

- VERSION, 350
- version number
 - ADPro, 350
- Vertical, 202
- vertical
 - aspect ratio, 32
- Vertical_Flip, 95, 269
 - ARexx control, 454
- VGA, 459
 - palette accuracy, 127
- viewer
 - for notes files, 512
- virtual screen, 29
- VLAB, 160
- VT_BUFFER, 216
- VUI, 32
 - selecting screen depth, 33
 - selecting screen mode, 33

W

- Window, 279
 - module name, 464
- window
 - adjusting size, 129
 - minimized, 129
- Workbench
 - font used, 129
 - starting ADPro, 86
 - starting FRED, 282

- starting Sentry, 497

- Workbench 2.04, 273
- Workbench 2.1, 273
- Workbench 3.0, 273
- WYSIWYG, 32

X

- XSIZE, 461

Y

- yellow correction, 214
- YSIZE, 461
- YUVN, 160

Z

- ZapDPI, 511

